A Framework for Batch Scheduling
with Variable Neighborhood Search
in Wafer Fabrication

der Bundeswehr
Universität München

A FRAMEWORK FOR BATCH SCHEDULING WITH VARIABLE NEIGHBORHOOD SEARCH IN WAFER FABRICATION

Robert Kohn

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Gutachter:

1. Gutachter: Prof. Dr. Oliver Rose
Institut für Technische Informatik
Fakultät für Informatik
Universität der Bundeswehr München

2. Gutachter: Prof. Dr. Stefan Pickl
Institut für Theoretische Informatik, Mathematik und Operations Research
Fakultät für Informatik
Universität der Bundeswehr München

Die Dissertation wurde am 11.12.2014 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 6.11.2015 angenommen. Die mündliche Prüfung fand am 6.11.2015 statt.

In effect, semiconductor manufacturing
seems to be an intellectual black hole -
despite any amount of brain power thrown at it,
it always offers challenges for doing it better.

(Leachman et al., 2002)

Abstract Die Halbleiterindustrie, als eine der größten und am schnellsten wachsenden Industrien der Welt, arbeitet kontinuierlich an der Reduktion ihrer Produktionskosten für konstant erschwingliche Produktpreise am Markt. Die Produktionslogistik spielt bei der Senkung der Produktionskosten in Halbleiterwerken (waferfabs) eine entscheidende Rolle. Man geht davon aus, dass Scheduling-Systeme in Kombination mit entsprechenden Optimierungsverfahren die derzeit eingesetzten Dispatch-Systeme als state-of-the-art Steuerungsverfahren in naher Zukunft ablösen werden. Insbesondere die Möglichkeit des Optimierens verschafft Scheduling-Systemen gegenüber Dispatching-Systemen entscheidende Vorteile. Viele Autoren teilen die Auffassung, dass exakte Optimierungsverfahren in Scheduling-Systemen nicht die erste Methode der Wahl zu sein scheinen. Stattdessen werden oft Metaheuristiken und lokale Suchverfahren zur Lösung von Scheduling-Problemen mit verschiedensten Randbedingungen herangezogen. Sie liefern akzeptable Lösungen mit vertretbarem Zeitaufwand.

Diese Arbeit beschreibt ein Scheduling-Framework und dessen Implementierung für den Einsatz als Batch-Scheduling System im Prozessbereich Diffusion/Oxidation in einer Halbleiterfabrik. Das Framework umfasst im Kern ein simulations-basiertes Scheduling-System mit Variabler Nachbarschaftssuche (VNS) zur Optimierung. 9 Kapitel beleuchten ausführlich die zugrundeliegenden theoretischen Hintergrnde und liefern gleichfalls wertvolle Erfahrungen aus der industrienahen Entwicklung eines Scheduling-Systems unter Praxisbedigungen.

Der theoretische Teil dieser Arbeit lässt sich zweiteilen. Zum einen stellt die ausführliche Literaturbersicht zum Thema Batch-Scheduling in der Halbleiterfertigung eine wesentliche Säule dieser Arbeit dar. Zum anderen vervollständigt die detaillierte Einführung in das Fachgebiet mit einer eingehenden Analyse der Komplexitätsklassen zu den wichtigsten Batch-Scheduling Problemen den Theorieteil dieser Arbeit.

Der Fokus des praktischen Teils beleuchtet Aspekte aus Industrie und Forschung gleichermaßen. Das implementierte System vereinbart diese beiden Anwendungsgebiete zu gleichen Teilen, in dem es einerseits der Forschung als Experimentiersystem dient und andererseits als Prototyp in der Industrie funktioniert.

Das Experimentiersystem eröffnet die Möglichkeit akademische Fragestellungen aus dem Bereich des metaheuristischen Batch-Schedulings zu beantworten. Die Experimente zeigen, dass auch kleine Änderungen in den Ausgangsbedingungen zu bedeutenden Änderungen in den Scheduling-Ergebnissen führen können. Es geht um die Frage, welchen Einfluss die Eigenschaften des Scheduling-Problems und die Parameter der Scheduling-Methode auf die Qualität der zu erreichenden Verbesserungen haben können. Darüber hinaus zeigen die Experimente welche dieser Parameter zur Optimierung von Ablaufplänen zu favorisieren sind.

Der Prototyp wurde gezielt für die speziellen Anforderungen der Industrie entwickelt. Das Design des Frameworks ermöglicht seine Nutzung als prototypisches Steuerungssystem im operativen Betrieb, als Real-Time Scheduling-System. Design, Implementierung und Test eines Scheduling-Systems sind anspruchsvolle Aufgaben. Die Installation eines solchen Steuerungssystems im laufendem Betrieb ist jedoch weitaus herausfordernder. Diese Arbeit beschreibt das implementierte Scheduling-System mit den wichtigsten Komponenten in den Bereichen Modellierung, Simulation und Optimierung, sowie die darunter liegenden Datenebenen mit Verbindung zum Fertigungsystem (MES) einer Halbleiterfabrik.

Key Words Batch Scheduling, Simulation-basiertes Scheduling, Variable Nachbarschaftssuche, Halbleiterfertigung

Abstract The semiconductor industry as one of the largest and fastest growing industries in the world needs to continuously reduce production costs to provide affordable products. Factory operations are likely to be major drivers to realize the necessary cost reductions in wafer fabrication facilities (waferfabs). For example operational scheduling systems powered by optimization techniques promise to replace dispatching systems as state-of-the-art control systems in the near future. Especially the capability of optimization makes scheduling systems superior to dispatching systems. Many authors share the opinion that exact optimization methods do not seem to be the method of choice in real-world scheduling systems. Instead metaheuristics and local search (LS) methods in particular are often used to solve scheduling problems with a variety of complicating constraints, since these algorithms can obtain good quality solutions within a reasonable time.

The core of this work revolves around the implementation of a scheduling framework developed to deploy it as an operational batch scheduling system in the diffusion and oxidation area of a waferfab. The implemented framework is essentially a simulation-based scheduling system powered by Variable Neighborhood Search (VNS). This thesis provides the underlying theoretical background and reports valuable practical experiences from implementing a scheduling system in a real-world industrial environment.

The focus of the theoretical part lies on two topics. First an extensive literature review about batch scheduling in wafer fabrication stands as one of the main pillars for this work. Secondly a detailed introduction to the batch scheduling topic with a detailed analysis of the complexity results of the most common batch scheduling problems completes the theoretical background of this work.

The focus of the practical work lies between the poles of academia and industry. The implemented framework is a balancing act between academia and industry since it basically comprises two systems: the experimental system and the prototype.

On one hand the experimental system offers the capability to properly investigate academic questions in the area of metaheuristic batch scheduling. The experiments show that even slight changes in the experimental setup can result in considerable changes of the output. The question is raised whether the problem instance's characteristics or the scheduling method settings have greater influence on the improvements. It is further shown by experiments how to ideally parametrize a VNS scheme optimizing schedules.

On the other hand the framework's prototype is purposefully designed and developed for the needs in industry. The intention of the framework's design is to provide a functioning prototype that is suitable to run as a real-time scheduling system on the operational level. Designing, implementing and testing a scheduling system is a demanding task, but deploying it in a waferfab that relies on dispatching to that date is even more challenging. This thesis describes the top-level scheduling system with all its modeling, simulation and optimization functionalities and the underlying data level connected to the waferfab's manufacturing execution system (MES).

Key Words Batch Scheduling, Simulation-Based Scheduling, Variable Neighborhood Search, Wafer Fabrication

Contents

List of Figures

List of Tables

Acronyms

ACO	Ant Colony Optimization
AGV	Automated Guided Vehicle
AHP	Analytical Hierarchical Process
AI	Artificial Intelligence
AIS	Artificial Immune System
AMHS	Automated Material Handling System
AMP	Adaptive Memory Programming
APC	Advanced Process Control
ASIC	Application Specific Integrated Circuit
ATC	Apparent Tardiness Cost
B&B	Branch and Bound
BATC	Batch Apparent Tardiness Cost
BCO	Bee Colony Optimization
BP	Batch Processing
BPM	Batch Processing Machine
BVNS	Basic Variable Neighborhood Search
C_{max}	Makespan
CAPA	Capacity
CFM	Continuous Flow Manufacturing
CFP	Conventional Furnace Processor
CFT	Continuous Flow Transport
CIM	Computer Integrated Manufacturing
CLOB	Character Large Object
CMOS	Complementary Metal Oxide Semiconductor
CMP	Chemical-Mechanical Polishing
CMS	Closed Machine Set
COP	Combinatorial Optimization Problem
CP	Constraint Programming
CR	Critical Ratio
CT	Cycle Time
CVD	Chemical Vapor Deposition
DBH	Dynamic Batching Heuristic
DES	Discrete Event Simulation
DJAH	Dynamic Job Assignment Heuristic
DP	Dynamic Programming
DRAM	Dynamic Random Access Memory
DSH	Dynamic Scheduling Heuristic
DTM	Deterministic Turing Machine
EA	Evolutionary Algorithm
EC	Evolutionary Computation
ECD	Electrochemical Deposition
EDA	Estimation of Distribution Algorithm
EDD	Earliest Due Date
EEPROM	Electrically Erasable Programmable Read-Only Memory
EFEM	Equipment Front End Module
EP	Evolutionary Programming
ERD	Earliest Release Date First
ERP	Enterprise Resource Planning

ERT	Earliest Release Time First
ES	Evolution Strategy
EST	Earliest Start Time First
FCFS	First Come First Served
FIFO	First In First Out
FO	Factory Operations
FOE	First-Only-Empty
FOUP	Front Opening Unified Pod
FPTAS	Fully Polynomial Time Approximation Scheme
GA	Genetic Algorithm
GEM	Generic Model for Communications and Control of Manufacturing Equipment
GLS	Guided Local Search
GP	Genetic Programming
GRASP	Greedy Randomized Adaptive Search Procedure
GUI	Graphical User Interface
GVNS	General Variable Neighborhood Search
HPC	High Performance Computing
IC	Integrated Circuit
IDM	Input Data Management
ILS	Iterated Local Search
IP	Integer Programming
ISO	International Organization for Standardization
ITRS	International Technology Roadmap for Semiconductors
KPI	Key Performance Indicator
L_{max}	Maximum Lateness
LAB	Look-Ahead Batching
LNS	Large Neighborhood Search
LP	Linear Programming
LS	Local Search
M&S	Modeling and Simulation
MBP	Mini-Batch Processing
MBS	Minimum Batch Size
MCR	Minimum Cost Rate Heuristic
MCS	Material Control System
MES	Manufacturing Execution System
MIP	Mixed Integer Programming
MOF	Metaheuristic Optimization Framework
MPC	Model Predictive Control
MPU	Micro Processing Unit
MRP	Material Requirements Planning
MRP II	Manufacturing Resources Planning
MS	Minimum Slack
NACH	Next Arrival Control Heuristic
NFLT	No-Free-Lunch Theorem
NTM	Non-Deterministic Turing Machine

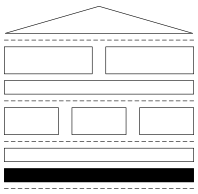
ODD	Operation Due Date
OHS	Overhead Shuttle
OHT	Overhead Hoist Transport
OHV	Overhead Hoist Vehicle
Op curve	Operating Curve
OR	Operations Research
OTD	On-Time Delivery
PAC	Probabilistic Approximate Completeness
PECVD	Plasma-Enhanced Chemical Vapor Deposition
PGV	Personnel Guided Vehicle
PSC	Planning, Scheduling and Control
PSO	Particle Swarm Optimization
PTAS	Polynomial Time Approximation Scheme
PVD	Physical Vapor Deposition
R&D	Research and Development
RCL	Restricted Candidate List
RGV	Rail Guided Vehicle
RHCR	Rolling Horizon Cost Rate
RTP	Rapid Thermal Processor
RVNS	Reduced Variable Neighborhood Search
SA	Simulated Annealing
SAT	Satisfiability Problem
SCM	Short Cycle Time Manufacturing
SCOP	Stochastic Combinatorial Optimization Problem
SDES	Stochastic Discrete Event Systems
SECS-I	SEMI Equipment Communications Standard Part 1
SECS-II	SEMI Equipment Communications Standard Part 2
SEM	Specific Equipment Model
SEMATECH	Semiconductor Manufacturing Technology
SEMI	Semiconductor Equipment and Materials International
SI	Swarm Intelligence
SPT	Shortest Processing Time
SS	Scatter Search
SVNS	Skewed Variable Neighborhood Search
SWP	Single-Wafer Processing
TA	Threshold Accepting
TCT	Total Cycle Time
THP	Throughput
TLNA	Time-Limited Next Arrival
TS	Tabu Search
TSP	Traveling Salesman Problem
TT	Total Tardiness
TU	Total Unit Penalties
TWCT	Total Weighted Cycle Time
TWD	Time Window Decomposition
TWT	Total Weighted Tardiness
TWU	Total Weighted Unit Penalties
UML	Unified Modeling Language

V&V	Validation and Verification
VLSN	Very Large Scale Neighborhood Search
VND	Variable Neighborhood Descent
VNDS	Variable Neighborhood Decomposition Search
VNS	Variable Neighborhood Search
VVA	Validation, Verification and Accreditation
WEDD	Weighted Earliest Due Date
WIP	Work in Process/Progress
WSPT	Weighted Shortest Processing Time

Symbols

$\alpha \mid \beta \mid \gamma$	A notation to classify deterministic scheduling problems by describing the specific characteristics of the scheduling problem in three fields.
$\mathcal{O}$	Worst-case time complexity function of an algorithm or a problem.
$\mathcal{P}$	Complexity class for decision problems that can be solved by a deterministic Turing machine in a polynomial time.
$\mathcal{NP}$	Complexity class for decision problems that can be solved by a non-deterministic Turing machine in a polynomial time.
$\propto$	Reducibility between optimization/decision problems in a polynomial bounded time.
$\stackrel{?}{=}$	Equality is unknown.

A Framework for Batch Scheduling with Variable Neighborhood Search in Wafer Fabrication



1 Introduction

Contents

The electronics industry has become one of the largest industries in the world and manufacturing of Integrated Circuits (ICs) is an important part of the global economy (Fowler and Rose, 2004; Mönch et al., 2011a). The semiconductor manufacturing industry with an annual sales of more than $250 billion worldwide at the end of 2007 is additionally considered as one of the fastest growing industries in the world (Mathirajan et al., 2010).

A high profitability of manufacturing organizations is simply put the result of two factors: high sales of products combined with low costs in manufacturing, at what low costs in manufacturing is the result of a high level of productivity (Hopp and Spearman, 2001). The semiconductor industry needs about a 30% cost per cm^2 reduction every 10 years to keep the costs and products affordable (Pillai, 2006). Since productivity improvements on the same wafer size have not yielded the necessary cost reductions, semiconductor industry historically has changed over to the next wafer size level approximately every 10 years. Today semiconductor manufacturer also focus on introducing a new generation of improved process equipment with greatly increased productivity in order to delay the next wafer size transition (Pillai, 2006).

In the past, most efforts to increase productivity and thus to reduce costs included *a*) decreasing the size of the chips, *b*) increasing the wafer sizes, and *c*) improving the yield. Beyond these three, considerable improvements of operational processes inside the semiconductor manufacturing system, also covered with the term Factory Operations (FO), today and in near future are likely to be the major drivers to realize the necessary cost reductions (Mönch et al., 2003, 2011a).

The International Technology Roadmap for Semiconductors (ITRS)[1] frequently provides a survey about recent and future developments in semiconductor industry (Cogez et al., 2007, 2011).

Key Performance Indicators (KPIs) Hopp and Spearman (2001) define basic performance indicators in a production system, alternatively referred to as Key Performance Indicators (KPIs). The Throughput (THP) of a production system is defined as the average output of the production process per unit time (e.g. parts per hour), also referred to as THP rate. An upper limit on the THP of a production system is its Capacity (CAPA). The utilization of a workstation can be seen as the fraction of time the workstation is used explicitly for production. It can be computed as the ratio of the arrival rate of the parts and its effective production rate. The effective production rate is defined as the maximum average rate at what the workstation can process parts (considering the effects of failures, setups, and all other detractors). The Work in Process/Progress (WIP) refers to the inventory (all the products) between the start and end points of a product routing, i.e. all the parts that have started but not have completed their processing. The Cycle Time (CT) is defined for a given routing and refers to the time a part needs from its release at the beginning of the routing until completing the last step at the end of the routing. Another basic goal of manufacturing is On-Time Delivery (OTD), that is meeting due dates that usually directly refer to customer orders.

Leachman and Hodges (1996) discuss a list of important metrics of semiconductor manufacturing performance with real-life data. They examine various measures for CT, OTD, yield and productivity. Leachman (2002) summarizes findings from benchmarking 10 200 mm manufacturing facilities. Montoya-Torres (2006) provides an overview on the most commonly used performance evaluation metrics in the semiconductor industry, mentioning relevant scientific works. The ITRS (Cogez et al., 2007) reports various benchmark values in semiconductor fabrication facilities, e.g. for CT and utilization. Godinho Filho and Uzsoy (2011) discuss dependencies among several KPIs.

[1] http://public.itrs.net/

Factory Operations (FO) As stated before, FO are likely to be major drivers to realize the necessary cost reductions (Mönch et al., 2011a). The primary goals of FO as a main pillar in manufacturing management are: *a*) minimal WIP, *b*) short CTs, *c*) maximum utilization of resources along with maximum THP, and *d*) high OTD (Hopp and Spearman, 2001); cf. (Cogez et al., 2007, 2011; Mönch et al., 2011a). Especially meeting due dates for optimal customer satisfaction along with short CTs have become one of the most decisive factors for the success of semiconductor manufacturers on the global market (Mönch et al., 2011a). The goal of production scheduling is to strike a profitable balance between these conflicting objectives (Hopp and Spearman, 2001; Rose, 2006).

Operating Curve (Op curve) Little (1961) formulates a fundamental law for manufacturing systems: the long-term average number of customers (L) in a stable system is equal to the long-term average effective arrival rate (λ) multiplied by the average time a customer spends in the system (W); or expressed algebraically: $L = \lambda W$. Given that the arrival rate is identical to the THP rate over time in a balanced system, Little's Law can also be stated as $WIP = THP \times CT$ with respect to the naming convention for manufacturing systems. Little's Law implies that reducing CT and reducing WIP is equivalent, provided that THP remains constant (Hopp and Spearman, 2001).

Aurand and Miller (1997) have developed the concept of the Operating Curve (Op curve) for manufacturing systems, further clarifying the relationship between the normalized CT (x-factor) and the machine utilization. Based on waiting line models from the Queuing Theory literature, the Op curve methodology incorporates the effect of variability typical for manufacturing processes. Hence, the Op curve provides a more clearer understanding of the relationship between the two fundamental indicators of factory productivity, i.e. the product CT and the asset utilization. The basic observation is that CT increases with an increasing utilization (respectively THP) and tends to infinity as THP converges to CAPA.

Refer to Figure 1 visualizing the concept of Op curve at hand of an example.

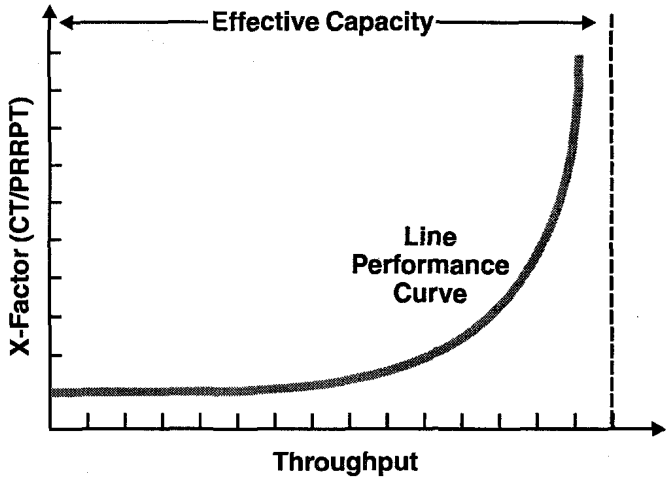


Figure 1: Line performance curve (Martin, 1998); cf. (Aurand and Miller, 1997)

Maximizing Throughput (THP) The capital cost of the production equipment runs in many billions of dollars and is 75% of the total factory capital costs, which drives the need to maximize the utilization of resources (Gupta et al., 2006; Pillai, 2006).

Maximizing the machine utilization means a higher return on investment due to higher a utilization of that capital equipment. The level of utilizing a manufacturing system can run with reasonable WIP and CT depends on the level of variability, knowing that 100% utilization is impossible. The higher the variability the manufacturing system shows, the lower is the utilization that has to be compensated (Hopp and Spearman, 2001).

Aurand and Miller (1997) coin the term Continuous Flow Manufacturing (CFM) covering the Op curve methodology proposed to benchmark and predict a manufacturing performance. One of

the principle methods used to improve manufacturing asset utilization is CFM. The key focus of CFM is to measure and manage the THP of machines by identifying and fixing problems in the factory. As stated before, CT increases with an increasing utilization (THP) and tends to infinity as THP converges to CAPA. This results in a trade-off between the costs of the reduced THP required to achieve a given CT and the value of the increased productivity achieved by this CT (Martin, 1998, 2000). The profitability in semiconductor manufacturing considerably depends on the ability to interactively manage the trade-off between these two KPIs (Fayed and Dunnigan, 2007). Essentially, increasing THP leads to smaller costs per wafer and reducing CT results in lower financial holding costs (Mönch et al., 2011a); cf. (Pinedo, 2005, 2008).

Minimizing Cycle Time (CT) Based on the observation that CT is a much more sensitive indicator of CAPA problems than THP, Martin (1998) suggests Short Cycle Time Manufacturing (SCM) to focus on CT measurements rather than on THP measurements. The reason is that CT increases rapidly as the THP approaches the effective CAPA, since CT and THP share a non-linear relationship as described with the Op curve. The central goal of SCM is to promote productivity improvements by reducing CAPA loss components, resulting in a shift of the Op curve that is tantamount to simultaneously improved THP and CT; cf. (Martin, 2000).

Reducing CTs directly *a*) leads to better responsiveness to the customer, *b*) maintains manufacturing flexibility, *c*) improves quality, *d*) reduces reliance on product demand forecasts, and *e*) leads to better product shipment forecasts (Hopp and Spearman, 2001). Beyond those positive effects on customer serviceability, CT has a significant impact on productivity learning (Martin, 1998). Usually it is said that the big companies eat the small, but today the fast run over the slow since the most important performance difference concerns speed in many aspects. Leading companies *a*) introduce new process technology earlier, *b*) qualify the technology faster, *c*) ramp up the yield and the volume more quickly, *d*) manage shorter CTs, and *e*) shorten process times on equipment with higher THP (Leachman, 2002). It is further taken into consideration that rapid CTs promote yield improvements, especially during the early stages of product and process life cycles (Wein, 1992; Cunningham and Shanthikumar, 1996). Moore (1998) describes an observation that was henceforward referred to as Moore's law. It says that the number of transistors in semiconductor devices doubles approximately every two years between subsequent technology nodes. Thus the number of chip layers continues to increase simultaneously. In order to prevent CT stagnation or even degradation as a result of an increasing number of chip layers, it is of particular importance to reinforce efforts to find breakthrough approaches that lead to continuous reductions in CT (Moore, 1998; Pillai, 2006); cf. (Cogez et al., 2007, 2011).

1.1 Motivation

Operations in a wafer fabrication facility (waferfab) account for more than 75% of the total CT and are also the largest component of costs within the value chain of semiconductor manufacturing (Mönch et al., 2011a). Scheduling (with sequencing rules) has a significant impact on the average CT, but has a considerable less impact on the factory performance compared to the effect of the chosen input control policy that is supposed to properly regulate the wafer starts in order to limit the WIP in the factory (Wein, 1988; Johri, 1993). The management of waferfabs has become increasingly interested in using effective production Planning, Scheduling and Control (PSC) techniques as a vehicle to achieve a competitive advantage (Gupta et al., 2006). Mönch et al. (2003) state that efficient PSC strategies currently create the best opportunity to improve operational processes in order to realize the necessary cost reductions; cf. (Mönch et al., 2011a). Major investments in PSC processes are made consistently by all major semiconductor manufacturing enterprises, including Intel and Samsung (Pinedo, 2008).

Modeling, Simulation and Optimization It is very likely that highly sophisticated computer-based techniques in the area of modeling, simulation and optimization will play a key role in the next-generation manufacturing systems. Underpinning this, six observations from the simulation viewpoint (Allen, 2011) and from the optimization viewpoint (Hansen et al., 2009) are summarized, which represent the main contributors to the continuous success of modeling, simulation and optimization:

a) continuing pressure for organizational efficiency, *b*) improved access to low-level data through new sensors and databases, *c*) enhanced visualization capabilities (realistic simulations), *d*) rapid improvement in computer performances, *e*) continuous research progress in theory and design of algorithms, and *f*) better communication of new ideas and integration in widely used complex software systems. These observations give strong indications that there is a pervasive need to use modeling, simulation and optimization for decision support, in particular for PSC techniques in current and future manufacturing systems (Fowler and Rose, 2004).

Dispatching versus Scheduling Dispatching systems set the de facto standard in scheduling and control of waferfabs, whereby which the APF RTD[2] product is installed in most waferfabs (Mönch et al., 2011a); cf. (Leachman, 2002).

Scheduling deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives (Pinedo, 2005, 2008). See (T'kindt and Billaut, 2006) for alternative definitions of the scheduling paradigm.

Fordyce et al. (2008) discuss the fundamentals of the paradigm shift from dispatching to scheduling in an IBM 300 mm fab, reporting substantial improvements in performance and significantly reduced overhead to adapt to changing circumstances. A dispatching system typically employs a set of simple priority rules, which are continuously further developed to complex rule-based decision systems that truly do a reasonable job. However, compared to scheduling systems, dispatching systems based on rules fundamentally lack a robust ability to: *a*) look across time, *b*) look across tools at a tool set, *c*) create an anticipated sequence of events at a tool set over some time horizon, *d*) establish a formal metric, and *e*) search alternatives.

Another reason stems from a continuous decrease of manual handling activities. With the shift from 200 mm to 300 mm wafer size, automated material handling generally replaces manual handling because of the increase in size and weight of the wafers. This fact additionally intensifies the need for scheduling approaches and drives the integration of scheduling solutions with automated material handling efforts taking the growing importance of automation into account (Chien et al., 2008; Mönch et al., 2011a). Refer to Section 3.3 for a short introduction into automated material handling in waferfabs.

In the past decades the standard wisdom was that the use of optimization in dispatch applications is technically not feasible. This has begun to change driven by continuous improvements in computing power and algorithms (Fordyce et al., 2008).

Technical Challenges Factors that generally complicate the scheduling process include: *a*) manufacturing complexity, *b*) randomness and variability, *c*) long time horizons, *d*) data inconsistency and availability, *e*) the lack of execution mechanisms, and *f*) the lack of a framework (Pinedo, 2008). From perspective of scheduling, a waferfab resembles a flexible job shop with various specific characteristics that make the scheduling process inherently very complicated and of critical importance (Pinedo, 2008). The operational control of semiconductor manufacturing facilities is a challenge, as these systems are among the most complex manufacturing environments encountered today (Gupta et al., 2006). Uzsoy et al. (1992b, 1994) give detailed descriptions of operational processes in semiconductor manufacturing facilities, focusing on challenges related to PSC issues. They highlight the following six factors that make PSC in the semiconductor industry particularly difficult: *a*) complex product flows (recirculation), *b*) random yields, *c*) diverse equipment characteristics, *d*) equipment downtime, *e*) production and development in shared facilities, and *f*) data availability and maintenance; cf. (Wein, 1988; Johri, 1993; Gupta et al., 2006; Shanthikumar et al., 2007) among others.

a) Complex product flows embody a high number of process steps, where a number of them perform on the same production equipment and thus each lot may repeatedly visits a machine multiple times. This kind of recirculation is also known as reentrant product flow. Even for a fixed process route, the material flow is highly dynamic due to disturbing effects related to scrap, on-hold, rework, and lot split/merge activities.

[2] http://www.appliedmaterials.com

b) Random (wafer) yields are subject to uncertain process yields that vary due to environmental conditions, i.e. problems with production equipment or material. Yield problems require a large amount of engineering hold time on both lots and equipment as part troubleshooting activities.

c) Diverse equipment characteristics are typically encountered in waferfabs. The characteristics of the equipment vary widely, e.g. in nature of batch processing machines, sequence-dependent setups, and the need for auxiliary resources.

d) Equipment downtimes state another critical issue since semiconductor manufacturing technology is extremely sophisticated. Semiconductor process equipment is often subject to unpredictable failures and requires extensive preventive maintenance and calibration efforts in order to maintain a certain level of process quality. Unpredictable equipment downtime is considered as the main contributor to uncertainty and variability in semiconductor manufacturing operations.

e) Production and development in shared facilities mirror the problem of efficiently reconciling production and engineering activities. Such facilities are also referred to as Research and Development (R&D) waferfabs. Driven by the need to continuously develop new products and processes, very often the same equipment is used for both production lots and engineering lots.

f) Data availability and maintenance is connected with extremely time-consuming activities tying considerable amounts of manpower. The sheer volume of data in a semiconductor manufacturing facility requires highly skilled capabilities in data management.

Additional complexity arises with continuous product diversification. Waferfabs with a higher product mix along with smaller lot sizes have become the norm, which makes the manufacturing even more complex due to an increased number of coexisting process and product flows. These so-called high-mix low-volume waferfabs find themselves confronted with profound impacts caused by smaller lot sizes. For example, if the standard lot size of 25 wafers is reduced to a seven wafer lot size across the entire fab, an 3.5× increase will occur in every lot-related activity in the factory. High-mix low-volume waferfabs in particular create a critical need to better schedule the production material in the factory in order to keep the fab running efficiently (Pillai, 2006). In (Cogez et al., 2007) the ITRS explicitly mentions the need to reduce losses from the high-mix effect. Mönch et al. (2011a) also note that scheduling is more complex in high-mix fabs.

Local Scheduling To our best knowledge, holistic waferfab-wide scheduling systems are not in use until today. The reason is that full-factory scheduling methods seem to be too computationally costly in comparison to dispatching methods. However, the lasting increase in computer efficiency drives full-waferfab scheduling methods more competitive. In a sense simulation-based scheduling is an intermediate concept between dispatching and true scheduling. Such scheduling systems based on detailed deterministic simulation models derive schedules from using dispatching rules, sometimes accompanied with search-based heuristics (Mönch et al., 2011a); cf. (Gupta and Sivakumar, 2002). Refer to Section 4.6.3 for a short examination of simulation-based scheduling approaches.

Given the complexity and the size of waferfabs, it is common to differentiate between fab-wide scheduling and detailed area scheduling (Chien et al., 2008). It is important to ensure consistency between these two levels, respectively between global and local scheduling decisions (Bureau et al., 2007). Scheduling a single machine group addressed to the single or parallel machine level has been demonstrated in some successful implementations. Mönch et al. (2011a) mention some scheduling approaches for specific work areas in semiconductor manufacturing, e.g. in the work areas related to diffusion/oxidation, lithography, and dry etch processes.

Batch Processing (BP) In today's leading waferfabs, most process and metrology equipment has become Single-Wafer Processing (SWP) tool configurations, but diffusion and oxidation as well as wet cleaning and some implantation operations are still subject to Batch Processing (BP) to a large extend. Such BP operations are typically performed in Conventional Furnace

Processors (CFPs). Refer to Section 3.2 for descriptions of common equipment types and Section 4.7 for equipment modeling approaches. Wafer fabrication involves numerous BP operations that considerably determine the system performance in terms of THP, WIP and CT (Fowler et al., 2002).

Historically, BP gained a significant output and cost efficiencies, but at the expense of long CTs. Since SWP enables a higher operating efficiency for small-lot sizes in high-mix low-volume waferfabs, there is a strong indication that BP will start to converge to SWP or Mini-Batch Processing (MBP) in the future fab (Pillai, 2006); cf. (Schmidt et al., 2006; Stubbe, 2010).

The costs of equipment in a new wafer fab is over 75% of the total factory capital costs (Gupta et al., 2006) and diffusion/oxidation processes are considered as the second costly process type, following the lithography process in the first place (Richards, 2013). A typical waferfab contains more than 70 different types of processing tools and the cost for a single machine can range between \$50,000 to \$10,000,000 (Kurz and Mason, 2008)

Depending on the product, the total number of BP steps in the process flow ranges from 50-100 steps (Pillai, 2006) and accounts for over 30% of the overall processing time (Sha et al., 2004, 2007). Compared to other process types, diffusion/oxidation operations at Batch Processing Machines (BPMs) come with long processing time requirements of about 10 hours or more (Johri, 1993; Mehta and Uzsoy, 1998; Mathirajan and Sivakumar, 2006b). For these reasons, the effective scheduling of diffusion/oxidation operations via BPMs is of particular importance for managing waferfab productivity (Mehta and Uzsoy, 1998).

1.2 Problem Description

A scheduling problem is basically a Combinatorial Optimization Problem (COP) with sequencing and/or partitioning decisions. The task is to find a feasible and optimal schedule for a number of jobs in a given machine environment. The resulting schedule is required to be feasible, i.e. the set of given constraints is satisfied without any exception so that the schedule is immediately executable on the shop floor. In contrast to the indispensable requirement of feasibility, optimality of a schedule is not necessarily required. However, the scheduling method always seeks for the optimal solution. The schedule will be regarded as optimal if no other feasible schedule with a better objective value exists.

Albers and Brucker (1993) define batching problems as combinations of sequencing and partitioning problems. The aspect of partitioning refers to the task of finding a partition of jobs into batches. The sequencing problem is given by the task to find an order of batches with an optimized objective value.

This work focuses on scheduling BP operations in the diffusion/oxidation area. Diffusion and oxidation operations in the waferfab frontend constitute typical scheduling problems with BPMs. Despite of the fact that from a technological point of view diffusion and oxidation do not belong to the same process group, since oxidation is a film formation process (see Section 3.1.1) and diffusion belongs to impurity doping processes (see Section 3.1.4), both are thermal processes typically performed by BPMs in the same area (even by the same machines). The wafers of usually six to twelve lots jointly undergo a diffusion/oxidation process in a cylindrical reactor (CFP). Among other constraints, the diffusion/oxidation process performed on BPMs is typically characterized by incompatible job families, i.e. jobs that belong to different families cannot be processed together (Chandru et al., 1993b; Uzsoy, 1995; Mehta and Uzsoy, 1998; Mathirajan and Sivakumar, 2006b).

A large quantity of variants of this particular type of a BPM scheduling problem is studied in literature. These variants usually define a simplified optimization model designed to mirror the decision problem that appears on the shop-floor under real-world conditions. These batch scheduling models enable researchers and practitioners to study the focused batch scheduling problem with respect to a defined set of circumstances, i.e. in different machine environments, under varying sets of constraints, and for several objective functions.

Graham et al. (1979) introduce the $\alpha \mid \beta \mid \gamma$-notation to classify deterministic scheduling problems by describing the specific characteristics of the scheduling problem in three fields: *a*) the machine environment (α-field), *b*) the constraints of the problem (β-field), and *c*) the optimality criterion (γ-field). Usually, the number of jobs is denoted by n, the number of machines by m, the subscript j refers to a job and the subscript i refers to a machine (Graham et al., 1979); cf. (Leung,

2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). The $\alpha \mid \beta \mid \gamma$-notation is used throughout the rest of this work.

1.2.1 Job Properties

With a scheduling problem, the given task is to find a schedule for a number of jobs in a given machine environment. A single job generally refers to a certain lot with some specific properties. Since this work is focused on parallel machine environments, a single job thereby refers to a single operation related to the lot that needs to process its next operation step according to its operation sequence on one of the parallel machines. Depending on the structure of the underlying scheduling model, a job may be characterized by a set of mandatory and optional parameters: *a*) a processing time (time span/continuous), *b*) a weight (continuous), *c*) a due date (date/time), *d*) a job size (integer), *e*) a release date (date/time), and *f*) a deadline (date/time); cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

An information regarding the processing time of a job is generally mandatory, whereas the weight, due date, job size, release date, and the deadline are considered as optional depending on the problem structure. A note on processing times in the $\alpha \mid \beta \mid \gamma$-notation is given only in the case of additional restrictions, e.g. the processing time is identical. A notion of non-identical job sizes, release dates and deadlines needs to be explicitly denoted in the β-field if existent. In contrast, due dates and weights are usually not specied in this eld only in the case of additional restrictions. Here, the objective function provides information about whether due dates or/and weights need to be considered.

Section 8.1.3 discusses experiments that investigate the effect of the number of jobs in a scheduling scenario.

Processing Time Generally, the processing time p_{ij} represents the processing time of job j on machine i; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

The diffusion/oxidation process is characterized by incompatible job families, where the process time of a batch is given by the job family, i.e. the process recipe on the machine. More specifically, a family of jobs is assigned to a process recipe on a machine and the process time of a batch is given by the running time of the related process recipe on that machine. Consequently all jobs of the same family have identical processing times on a particular machine, but family processing times may differ between machines due to different process recipes specifically designed for a particular machine; cf. (Chandru et al., 1993b; Uzsoy, 1995; Mehta and Uzsoy, 1998; Mathirajan and Sivakumar, 2006b). A note on processing times in the $\alpha \mid \beta \mid \gamma$-notation is given only in the case of additional restrictions, e.g. when the processing time is identical.

The processing time for a typical diffusion/oxidation process performed in a CFP takes four to eight hours, but can also take more than ten hours; cf. (Johri, 1993; Mathirajan and Sivakumar, 2006b).

Loading and unloading a CFP with six to twelve lots takes a considerable amount of time, roughly 30 minutes depending on the machine and the handling system. The time spent for loading and unloading obviously depends on the number of jobs, but may be sufficiently considered as a constant time accruing between processing two consecutive batches on a machine. From perspective of modeling and scheduling load and unload times may be included in the processing time. A typical CFP is equipped with a single reactor and thus is not capable to run a process during (un)loading the machine; cf. (Yugma et al., 2008; Klemmt et al., 2011).

But there exist variants of CFPs with two or three reactor tubes. This type of CFP runs more than one process simultaneously and thus offers to load and/or unload the machine while it is processing; cf. (Johri, 1993).

Section 8.1.5 discusses experiments that investigate the effect of different distributions of processing times in a scheduling scenario.

Weight The weight w_j of job j is a priority factor reflecting the importance of the job in relation to the other jobs in the system; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). Lots with higher priorities, sometimes called rocket lots or hot lots, usually refer to urgent customer orders or important engineering activities. The existence of weights is not explicitly specified in

the β-field, but the objective function provides information about whether weights need to be considered.

Section 8.1.10 discusses experiments that investigate the effect of different numbers of priority classes and weighting schemes in a scheduling scenario.

Due Date The due date d_j of job j represents the date the job is expected or planned to complete its process, e.g. the shipment date committed to the customer; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). It is common practice in waferfabs to set Operation Due Dates (ODDs) for a lot once it enters the manufacturing system; cf. (Rose, 2002, 2003b,a). An ODD refers to a precalculated due date assigned to each operation of a lot according to a product-specific process sequence. Similar to job weights, the existence of due dates is not explicitly specified in the β-field, but the objective function provides information about whether due dates need to be considered.

Section 8.1.9 discusses experiments that investigate the effect of different initial due date settings in a scheduling scenario.

Job Size The standard lot size of 25 wafers is given by the maximum number of wafers the carrier can hold. In most cases, a lot is filled to the maximum and carries 25 wafers. In fact, the number of wafers in the lots differ due to a small-lot manufacturing policy or caused by accidental effects that have lead to a removal of one or more wafers. The number of wafers, respectively the job size is usually denoted with s_j. The existence of non-identical job sizes needs to be explicitly noted in the β-field. In this case, the batch scheduling problem is additionally constrained with non-identical job sizes. If this note is not existent in the β-field, all jobs will be assumed to have the identical size.

Section 8.1.8 discusses experiments that investigate the effect of different dedication schemes with varying density factors in a scheduling scenario.

Release Date The release date r_j of job j is the time the job arrives at the system, also referred to as the ready time or arrival time. A job j can start its processing earliest at r_j. If this symbol is not present in β-field, then the processing of job j can start at any time, i.e. all the jobs are available at time zero; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

Section 8.2.2 and Section 8.2.3 discuss experiments that investigate the effect of predicted job arrivals in a scheduling scenario.

Deadline The deadline $\bar{d}_j$ of job j represents the latest time at which job j must be completed; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). The existence of deadlines needs to be explicitly mentioned in the β-field. If the symbol $\bar{d}_j$ is present, then the jobs will be subject to deadline constraints. Deadlines are also referred to as maximum time lags, time boundaries, or time constraints (Yugma et al., 2008; Klemmt et al., 2011; Mönch et al., 2011a).

Completion Date The completion date refers to the time when the job finishes its processing on the machine. With respect to a schedule, the completion date of job j is usually denoted with C_j. The objective to be minimized is always a function of the completion times of the jobs; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

1.2.2 Machine Environment

The α-field refers to the machine environment that basically defines the manufacturing system underlying the scheduling problem. The simplest case refers to a manufacturing system with a single machine only. The notion $1|\cdot|\cdot$ refers to a single machine scheduling problem, whereby the symbol $\cdot$ stands for an unspecified field. Since manufacturing systems rarely consist of a single machine, there exists a set of templates for machine environments represented in the α-field, e.g. variants of open shops, job shops, flow shops, and parallel machines; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

A waferfab is typically organized in a number of work areas, given that each work area covers a set of similar machines. In this context, the term similar roughly refers to the architecture of

the machines as well as to the type of processes they provide. There exists a strong relationship between the machine type and the process type since a specific process is often performed by a specific type of machine (see Table 20).

For the entire machine pool in the fab, there exists a number of subsets of machines so that each subset provides a disjunctive set of processes, also referred to as Closed Machine Sets (CMSs) or work centers. Based on the shop floor layout, the machines of a single work center are often located in near proximity, but may be also locally distributed on the shop floor. Section 3.4 briefly examines shop floor layouts. For a typical waferfab such a CMS roughly counts up to several dozens of machines that form a manufacturing entity from the viewpoint of production logistics. In most cases a lot receives the process of succeeding operations in different work centers; cf. (Mönch et al., 2011a).

Section 8.1.2 discusses experiments that investigate the effect of number of machines in a scheduling scenario.

Parallel Machine Environments Most definitions of work centers in waferfabs represent typical parallel machine scheduling problems, which are suitable to be separately considered and solved as an independent system. In the basic parallel machine scheduling model, a job proceeds four basic activities: *a*) entering the system, *b*) waiting for processing (queuing), *c*) performing the process on a machine, and *d*) leaving the system.

The concept of parallel machines describes a manufacturing system consisting of a number of machines in parallel where each job in the case of machine eligibility restrictions requires a single operation on any of the parallel machines, or on a subset of them. Among parallel machine scheduling problems, scheduling literature commonly differentiates between three types of parallel machines: *a*) identical machines (Pm), *b*) uniform machines (Qm), and *c*) unrelated machines (Rm). The parallel machine environments Pm, Qm, and Rm differ in their processing speed concepts. Identical machines (Pm) have the same speed and a job needs the time p_j on every machine. Uniform machines (Qm) have different speeds s_i and a job j needs $\frac{p_j}{s_i}$ time on machine i. Unrelated machines (Rm) can process different jobs at a different speed, i.e. machine i can process job j at speed s_{ij} and a job j spends $\frac{p_j}{s_{ij}}$ time on machine i for processing; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

Diffusion/Oxidation Area Despite of the fact that the machines of a single work center perform identical or at least similar processes, the machines may belong to different tool types and thus show different characteristics. It is even be conceivable that a diffusion/oxidation work center has a mixed structure in a sense that fundamentally different machines form the CMS, performing both SWP and BP.

In this work, it is assumed that a diffusion/oxidation work center under study exclusively consists of BPMs, i.e. CFPs. However, the machines may differ in their dedicated processes, processing times, and batch sizes. The typical diffusion/oxidation process in a furnace is characterized by incompatible job families, where a family of jobs is assigned to a process recipe on a machine and the process time of a batch is given by the running time of the related process recipe on that machine.

Since the family processing times may differ between machines due to different process recipes specifically designed for a particular machine, the corresponding scheduling problem embodies unrelated machines (Rm) and can be denoted with $Rm \mid \cdot \mid \cdot$.

This work is dedicated to the scheduling problem with BPMs in the diffusion/oxidation area; The implemented framework as well as the documented experiments exclusively deal with a parallel unrelated BPMs environment. However, this work gives an extensive review of BPM scheduling problems for single and parallel machine environments under varying constraints in Section 2.

1.2.3 Constraint Environment

The β-field further refines the scheduling problem by defining a set of constraints to be considered; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). It may contain multiple entries or no entry at all.

Pinedo (2005, 2008) distinguishes between hard and soft constraints. A hard constraint has to be satisfied at all costs. A soft constraint basically refers to a preference. If any of the given hard constraints is violated, the schedule will no longer be feasible and therefore not executable on the shop floor. In this context, soft constraints basically refer to a kind of an objective function that incorporates penalty costs in the case of violated soft constraints.

Generally, a scheduling problem is characterized by an arbitrary set of constraints depending on the structure of the underlying model. The typical BPM scheduling problem in waferfabs is bounded in a sense that the maximum batch size is limited by the maximum capacity of the BPM. The typical (bounded) BPM scheduling problem in the diffusion/oxidation area is further subject to *a*) non-identical job sizes, *b*) incompatible families, *c*) release dates, and *d*) machine eligibility constraints. This work studies numerous variants of BPM scheduling models under varying sets of these constraints. Moreover, the studied problem is further known to typically be subject to deadlines and machine unavailability constraints, also referred to as machine breakdowns. However, this work does not explicitly study the effects of breakdowns or deadlines. With the exception of those mentioned constraints that typically appear in BPM models for diffusion/oxidation processes, this work omits *a*) precedence constraints, *b*) preemptions, *c*) sequence dependent setup times, *d*) recirculation, *e*) job splitting, and *f*) rejections among other constraints, which are not typical for the focused problem, but are reported to be present in other BPM models (compare the literature review in Section 2).

Parallel Batching (*p-batch*) The scheduling problem present at diffusion/oxidation furnaces is obviously a typical parallel batch scheduling problem since the wafers of batched jobs simultaneously undergo a thermal process inside the furnace reactor. Consequently the jobs in a batch have the identical starting, processing, and completion time.

Brucker (2007) defines a parallel batch as a grouped set of jobs jointly processed on the same machine and introduces the abbreviation $p-batch$ that emerged to the widely accepted notion for parallel batching problems in scheduling literature. Thus, the notion $p-batch$ in the β-field further refines the machine environment in a sense that parallel BP is allowed; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). The literature review in Section 2 exclusively covers (bounded) $p-batch$ problems.

Batch Size Constraints ($b<n$) Generally it is distinguished between bounded and unbounded $p-batch$ problems. The (maximum) size of a batch is commonly denoted with the parameter b, whereby the bounded case $b < n$ refers to those $p-batch$ problems in which the batch size is limited by the machine capacity; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

With respect to the focused $p-batch$ problem in the diffusion/oxidation work area, it is required to consider two types of batch size limits related to the machine capacity: the number of jobs/lots and the number of wafers. Refer to Section 4.7.3 that introduces a general modeling concept for waferfab production equipment; cf. (Kohn et al., 2010). On one hand, there exists a constraint on the maximum number of lots in a batch; the maximum batch size in terms of jobs/lots is usually limited by the number of load ports provided by the machine. On the other hand, the maximum number of wafers in a batch is limited by the number of internal wafer slots offered by the furnace reactor. A typical vertical batch furnace in a 200 mm waferfab with a standard lot size of 25 wafers is equipped with eight load ports and provides space for up to 200 wafers. The maximum batch sizes, both the wafer capacity limit and the maximum number of jobs, may differ between machines due to different machine architectures and even between job families due to different process recipe specifications. Clearly, neither of both limits is allowed to be exceeded by any batch; cf. (Yugma et al., 2008; Klemmt et al., 2011).

In the majority of research focused on $p-batch$ problems in the diffusion/oxidation area of waferfab frontends, the maximum batch size b simply refers to the maximal number of jobs/lots allowed for grouping for a batch. It is assumed that the job sizes are equal, i.e. the number of wafers is identical among all the lots. The maximum batch size b in terms of jobs implicitly represents the limit for the wafer capacity of the furnace reactor. More precisely, the maximum number of jobs b is the ratio of the maximum wafer capacity and the standard lot size and b is assumed to be equal to the number of load ports provided by the machine. Using the $\alpha \mid \beta \mid \gamma$-notation the basic bounded $p-batch$ problem can be described with $\cdot \mid p-batch, b < n \mid \cdot$.

But most practitioners probably would consider ignoring the existence of non-identical job sizes as an insufficient simplification. For some equipment types, the maximum job limit is even greater than the number of load ports. This kind of machines offer a special reloading operating mode in which it is possible to load the machine with a number of additional lots after transferring the wafers of the lots on the load ports into the reactor. In this case, an empty carrier (its wafers are already transferred into the reactor) on a load port is replaced by a filled one, which is then unloaded to the reactor before the process begins. The motivation behind this special loading policy is to improve managing the utilization of those BPMs in the presence of a non-negligible amount of partially filled lots in the system. The importance of differentiating between both types of batch size limits in the design of $p-batch$ scheduling models for the diffusion/oxidation area is amplified by the current trend to manufacture small lot sizes. The literature review in Section 2 exclusively covers bounded $p-batch$ problems. Section 8.1.6 discusses experiments that investigate the effect of different batch sizes in a scheduling scenario.

Non-Identical Job Sizes (B, s_j) As stated before, the number of wafers in the lots may differ. In this case, the scheduling problem is additionally constrained with non-identical job sizes, which imply that the sum of the wafers in a batch is limited by the maximum wafer capacity of the machine. In this context, the maximum wafer capacity of the machine is denoted with B and the size of a job with s_j, whereupon both B and s_j need to be mentioned in the β-field.

Originally, the typical $p-batch$ problem characterized by non-identical job sizes stems from heat-treatment operations present in waferfab backends, also referred to as burn-in operations. Depending on the product type, the jobs may require a different number of boards that define the size of the job. The capacity of the oven is given by the maximum number of boards it can accommodate. This kind of scheduling problem is typically described with $\cdot \mid p-batch, B, s_j \mid \cdot$; cf. (Kempf et al., 1998; Mathirajan and Sivakumar, 2006a).

The notion $\cdot \mid p-batch, B, s_j \mid \cdot$ applied for the focused diffusion/oxidation model would imply that there effectively exists no load port limit, i.e. the machine provides a reloading operating mode that allows the attached loading system to continue reloading additional lots until the wafer capacity of the machine is reached.

To our best knowledge, the reloading operating mode is limited to a small number of additional lots. Hence, the notion $\cdot \mid p-batch, b<n, B, s_j \mid \cdot$ is regarded as a proper description for the diffusion/oxidation model with respect to machine capacity constraints. Following this notation, b denotes the maximum number of jobs in a batch, B denotes the maximum number of wafers in batch, and s_j implies that the jobs are subject to non-identical sizes. The literature review in Section 2 covers $p-batch$ problems with and without non-identical job sizes. Section 8.1.8 discusses experiments that investigate the effect of different dedication schemes with varying density factors in a scheduling scenario.

Incompatible Job Families ($fmls$) The $p-batch$ scheduling problem present in the oxidation/diffusion area typically deals with incompatible job families ($fmls$). A job j belongs to a certain job family and jobs from different families cannot be batched together. Jobs from different families are incompatible since a job family relates to a certain process specification that defines temperature, gas mixture and running time among other parameters in order to achieve a well-defined chemical reaction. Moreover, a family of jobs is assigned to a process recipe on a machine and the process time of a batch is given by the running time of the related process recipe on that machine. Consequently all jobs of the same family have identical processing times on a particular machine, whereby family processing times may differ between machines due to different process recipes specifically designed for a particular machine; cf. (Chandru et al., 1993b; Uzsoy, 1995; Ghosh and Gupta, 1997; Mehta and Uzsoy, 1998; Leung, 2004; Mathirajan and Sivakumar, 2006b; Pinedo, 2008).

The notion $fmls$ in the β-field refers to the constraint of incompatible job families. If the notion $fmls$ is missing in the beta-field, then any pair of jobs will be compatible and can be batched together. The basic bounded $p-batch$ scheduling model with incompatible families can be described with $\cdot \mid p-batch, b<n, fmls \mid \cdot$.

The literature review in Section 2 covers $p-batch$ problems with and without incompatible families. A special form of batch incompatibility constraints between jobs is given by graph

compatibilities (Boudhar, 2003; Finke et al., 2008). The Section 8.1.4 discusses experiments that investigate the effect of different numbers of job families in a scheduling scenario.

Machine Eligibility Restrictions (M_j) Machine eligibility restrictions (M_j) further refine the machine environment with process dedications. A job j cannot be processed on any machine, but only on any one belonging to a specific subset M_j. The set M_j denotes the set of machines that can process job j. If the β-field does not contain M_j, job j can be processed on any one of the available machines as defined in the default model of scheduling problems; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

In the context of the focused batch scheduling problem present in the diffusion/oxidation area, machine eligibility restrictions manifest themselves in a mapping between job families and machines, i.e. each job family is dedicated to one or more machines. More specifically, each job family is linked to a process recipe that is qualified on a machine. Vice versa, an equipment is qualified for a set of process recipes and thus offers processing for the corresponding set of job families. Considering the fact that each job belongs to a job family, there exists a subset M_j of machines that is capable of processing each job; cf. (Yugma et al., 2008; Klemmt et al., 2008, 2011).

Qualifying a process on a machine means to ensure that the desired process reliably performs within a predened range of process parameters. By doing several tests it is possible to find out whether the thickness of an oxide lm lies in-between a certain range after completing the process for example. Qualifying a process on a machine can be very time- and resource-consuming, but increases the flexibility of a work center and has a positive influence on the Op curve; cf. (Fayed and Dunnigan, 2007; Johnzén et al., 2007, 2008, 2011). The terms equipment qualification, equipment dedication and process dedication are often used synonymously in this context.

As mentioned before, the goal is to schedule a set of parallel machines, which typically form a CMS qualified for a set of processes that is disjunctive to other process subsets from other CMSs; cf. (Johri, 1993). The particular mapping between machines and qualified processes within a CMS can be formulated as a matrix of machines and qualified processes on these machines, also referred to as dedication matrix. The dedication matrix is characterized by a density factor. A dedication density factor that equals one would describe homogenous CMS in which every job can be processed on any machine, i.e. no restrictions exist. A factor below one would represent an inhomogeneous CMS implying that a certain number of machines does not provide all the entire set of processes available. Likewise, this work correspondingly deals with a matrix of machines and job families in the scheduling model described.

A typical bounded $p-batch$ scheduling problem with a machine eligibility restriction can be suitably described with the notion $\cdot \mid p-batch, b<n, M_j \mid \cdot$. A mapping M_j between jobs and machines can be easily deduced from the dedication matrix between job families and machines and from the membership of jobs to job families.

Beyond process dedications, the set of allowed machines M_j may be further constrained for a job due to several reasons, e.g. strict policies in quality management or material flow control. For example, additional constraints, set permanently or temporarily, may purposefully prohibit the processing of a job only at a defined operation on a certain machine, or based on other combinations of job attributes. Especially R&D waferfabs deal with several systems on several system layers in order to allow or prohibit distinct process operations. In contrast, a job may be explicitly dedicated to be processed on a single predefined machine at a certain operation, e.g. as a result of Advanced Process Control (APC). With respect to diffusion/oxidation furnaces it is required to regularly monitor the quality of the process by adding a number of non-productive wafers to a regular product-batch. In this special case a job with control wafers is created and dedicated to a distinct process on a machine.

However, finally each job is allowed to be processed on a set of machines M_j. Several authors include machine eligibility constraints in their parallel batching models; cf. (Klemmt et al., 2008, 2011; Li and Qiao, 2008; Yugma et al., 2008; Li et al., 2009b).

Section 8.1.7 discusses experiments that investigate the effect different of dedication schemes with varying density factors in a scheduling scenario.

Release Dates (r_j) The release date r_j of job j describes the time when the job j enters the system, also referred to as ready time or arrival time. A job j can start its processing earliest at r_j;

cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). If this symbol is not present in β-field, then the processing of job j can start at any time, i.e. all the jobs will be available at time zero. The notion $\cdot \mid p-batch, b<n, r_j \mid \cdot$ refers to a bounded scheduling problem with dynamic arrivals.

The way researchers examine a deterministic parallel machine scheduling problems with arrivals is in a sense detached from reality. First the focused parallel machine environment is simplified to an independent manufacturing system though it is usually embedded into to a larger complex manufacturing system, e.g. a work center in the diffusion/oxidation area is part of a waferfab. Second, it is simply assumed that the arrival date is known, not considering that reliably predicting arrival dates in a waferfab is a complex task. Refer to Section 4.6.1 for further information about prediction and forecasting methods.

The existence of release dates in a scheduling model implicitly presupposes that a suitable system for job arrival date predictions exists (Yugma et al., 2008). The capability of predicting job arrivals first includes a suitable equipment model used to predict completion dates of currently running jobs. Creating and maintaining equipment models that allow accurate completion date predictions in short time is itself a tough task. This task will become even more difficult if it is intended to predict completion dates of queued jobs at the preceding operation steps in addition to currently running jobs. At this point short-term simulation models of high quality need to be deployed.

After a job has completed its process at the preceding operation, and after unloading from the machine, usually an Automated Material Handling System (AMHS) transfers the lot to its destination, e.g. a material storage equipment located in a work area of the succeeding operation. Once the predicted completion date at the previous operation is known, the arrival date to predict additionally needs to incorporate transport and handling times that accrue due to AMHS activities.

The literature review in Section 2 covers $p-batch$ problems with and without release dates. Section 8.2.2 and Section 8.2.3 discuss experiments that investigate the effect of predicted job arrivals in a scheduling scenario.

Deadlines ($\bar{d}_j$) The deadline $\bar{d}_j$ of job j represents the latest time at which job j must be completed. The existence of deadlines needs to be explicitly mentioned in the β-field; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). The notion $\cdot \mid p-batch, b<n, d_j \mid \cdot$ refers to a bounded scheduling problem with deadlines. Deadlines are also referred to as maximum time lags, time boundaries, or time constraints; cf. (Yugma et al., 2008; Klemmt et al., 2011; Mönch et al., 2011a).

Such time bounds typically occur between the wet etch/clean and furnace operations, but also appear between other operations. The motivation behind is driven by quality concerns; the idea is to prevent unintentional oxidation and contamination effects that might occur during waiting for the next process. In the case of violated time bounds, the corresponding lot might be scrapped, but at least needs an additional inspection and/or some sort of rework, e.g. has to visit a preceding cleaning step again; cf. (Johri, 1993; Scholl and Domaschke, 1999, 2000; Klemmt and Mönch, 2012). Several authors include deadlines in their parallel batching models; cf. (Ikura and Gimple, 1986; Mönch et al., 2006a; Klemmt et al., 2008; Yugma et al., 2008; Bar-Noy et al., 2009; Klemmt et al., 2011; Koehler and Khuller, 2013).

Machine Breakdowns ($brkdwn$) In the case of machine breakdowns ($brkdwn$) we face a situation in which a number of machines may not be continuously available during the focused time period. The machine is not available for regular processing due to scheduled or unscheduled maintenance, or it is hold busy with engineering activities. The existence of periods in time where machines are not available for processing is also referred to as machine availability constraints; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). The notion $\cdot \mid p-batch, b<n, brkdwn \mid \cdot$ refers to a bounded scheduling problem with breakdowns.

In a deterministic scheduling model the periods of unavailability are known in advance. Since preemption is not allowed, a process batch is required to be completed before a period of unavailability begins. Clearly, a process batch starts earliest after a breakdown period ends. Yugma et al. (2008) and Klemmt et al. (2011) report the inclusion of machine breakdowns in their parallel batching models.

1.2.4 Scheduling Objectives

The top priority of any manufacturing organization is to gain a high profitability. A high level of profitability relies on a number of subordinated performance indicators (Hopp and Spearman, 2001). The most important are: *a*) throughput (THP), *b*) cycle time (CT), and *c*) on-time delivery (OTD); cf. (Mönch et al., 2011a).

In order to quantify these performance indicators, there exist numerous performance measures that can serve as optimality criteria for scheduling problems in form of objective functions. The γ-field refers to the optimality criterion chosen for the scheduling problem at hand. In practice, the optimality criterion given by an objective function is often a composite of several basic performance measures. In this case, the scheduling problem becomes a multi-objective optimization problem. However, the objective to be minimized is always a function of the completion times of the jobs; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). In the following important performance measures for scheduling problems are introduced, in particular those that are of interest for batch scheduling problems.

Section 8.1.11 and Section 8.1.12 discuss experiments that investigate the the effect of different objective functions and their relationships in a scheduling scenario.

Throughput (THP) Maximizing THP is often of utmost importance in many manufacturing facilities. The THP of an entire facility is limited by its bottleneck machines, i.e. those machines in the facility that have the lowest CAPA. The objective Makespan (C_{max}) is closely related to the THP objective. C_{max} is equivalent to the completion time of the last job that leaves the system, defined as $max(C_1, \ldots, C_n)$, where C_j is the completion time of job j. Optimizing C_{max} is of particular importance when there is a finite number of jobs. Minimizing the C_{max} in a parallel machine environment implies balancing the work load over the various machines, which also leads to a higher THP rate when there is a constant flow of jobs over time. Since THP and the utilization are strongly connected, a shorter C_{max} in turn implies a higher utilization; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). Minimizing C_{max} results in larger values of THP corresponding to a higher level of utilization, which in turn leads to smaller costs per wafer (Mönch et al., 2011a).

Another THP-related performance measure that characterizes BP performance is the batching coefficient. For a particular machine over a given time horizon, the batching coefficient is the ratio of the average number of jobs in a batch and its maximum batch size (Yugma et al., 2008).

Cycle Time (CT) Improving CT is supposed to come with various positive effects, e.g. improved yield (Wein, 1992; Cunningham and Shanthikumar, 1996), productivity learning (Martin, 1998), and customer serviceability (Hopp and Spearman, 2001). The CT refers to the time a job spends in a system until its processing is completed. The terms completion time, flow time, throughput time, and lead time altogether most often synonymously refer to the term CT that is preferably used in this work. The completion date C_j refers to the time when the job j finishes its processing on a machine. With respect to a schedule, the completion date C_j also refers to the (CT) of the job j.

There are two important objective measures: the Total Cycle Time (TCT) and the Total Weighted Cycle Time (TWCT). The TCT is defined as the sum of the completion times of the jobs, denoted by $\sum_{j=1}^{n} C_j$. The TWCT represents the weighted counterpart of $\sum_{j=1}^{n} C_j$, defined as the sum of weighted completion times of the jobs, denoted with $\sum_{j=1}^{n} w_j C_j$. Minimizing $\sum_{j=1}^{n} C_j$ and/or $\sum_{j=1}^{n} w_j C_j$ results in lower total holding or inventory costs incurred by the schedule; cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

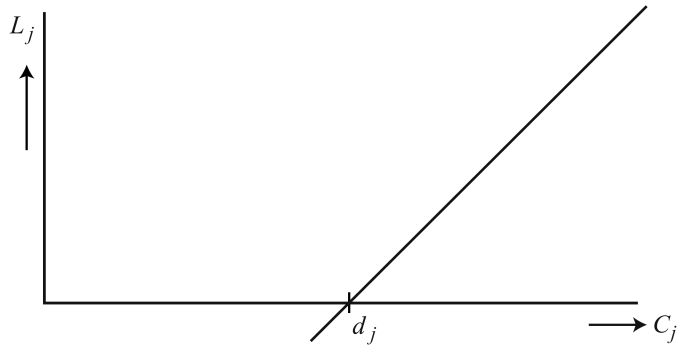
Another objective is given by the normalized CT (x-factor), originally defined in (Aurand and Miller, 1997). The x-factor is defined as the ratio of the completion time of a job and its processing time on the machine. Since the completion time is composed of the waiting/queuing time and the processing time, the x-factor also represents the ratio between the waiting time and the processing time. Yugma et al. (2008) employ the x-factor as a vehicle to minimize the CT.

On-Time Delivery (OTD) There are several important objectives that are related to due dates intended to gain a high level of OTD, which in turn increases customer satisfaction. Typical OTD

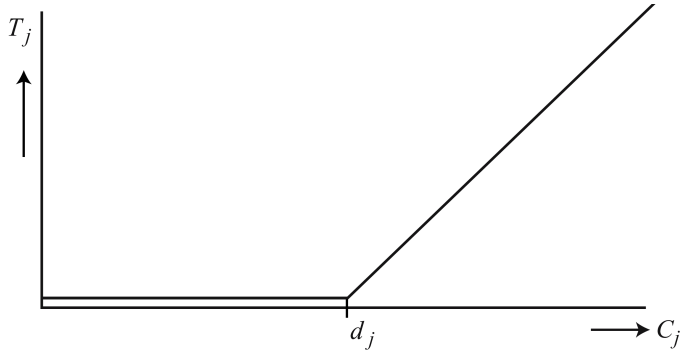
performance measures are *a*) Maximum Lateness (L_{max}), *b*) Total Unit Penalties (TU), and *c*) Total Tardiness (TT); cf. (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

L_{max} is defined as $max(L_1, \ldots, L_n)$ where the lateness of job j is then $L_j = C_j - d_j$, given that d_j denotes the due date and C_j the completion date of job j. Minimizing L_{max} is in a sense equivalent to minimizing the worst performance of the schedule. TU refers to the total number of tardy jobs is given by the sum of unit penalties incurred by the schedule, denoted with $\sum_{j=1}^{n} U_j$. The unit penalty U_j of job j is defined as $U_j = 1$ if $C_j > d_j$; otherwise $U_j = 0$. The weighted number of tardy jobs refers to the sum of Total Weighted Unit Penalties (TWU), denoted with $\sum_{j=1}^{n} w_j U_j$, given that the different jobs carry different priority weights w_j. This objective does not focus on the question how tardy a job actually is, but on the question whether it is tardy or not. However, minimizing $\sum_{j=1}^{n} U_j$ may result in practically unacceptable schedules with some jobs being very tardy. This concern is addressed with the due date related objective TT. The TT is defined as the sum of all tardiness values ($\sum_{j=1}^{n} T_j$) where the tardiness T_j of job j is defined as $max(C_j d_j, 0)$, which is identical to $T_j = max(L_j, 0)$. The Total Weighted Tardiness (TWT) ($\sum_{j=1}^{n} w_j T_j$) represents the weighted counterpart of $\sum_{j=1}^{n} T_j$, given that the different jobs carry different priority weights w_j.

Refer to Figure 2 for a graphical explanation of the definitions of lateness and tardiness.



(a) The lateness L_j of job j



(b) The tardiness T_j of job j

Figure 2: The lateness and the tardiness of a job (Pinedo, 2005, 2008)

1.3 Methodology

Dispatching systems set the de facto standard in scheduling and controlling waferfabs (Mönch et al., 2011a); cf. (Leachman, 2002). In the past decades the standard wisdom was that the use of optimization in dispatch applications is technically not feasible. This has begun to change, driven by continuous improvements in computing power and algorithms. Especially the capability of optimization makes scheduling systems superior to dispatching systems, respectively the ability to search alternatives combined with a formal metric in terms of an objective function used to compare alternative decisions. Today the real barrier is of cultural and not technical nature (Fordyce et al.,

2008).

Pinedo (2004, 2005, 2008) first and foremost distinguishes between two types of scheduling problems: deterministic scheduling and stochastic scheduling problems. Stochastic scheduling usually deals with simple priority or dispatching rules providing immediate decisions in a real-time control system. Deterministic scheduling emphasizes the aspect of optimization. Deterministic scheduling models are used to evaluate the performance of optimization methods in solving scheduling problems as a special form of COPs with the goal of minimizing or maximizing a given objective function. Since deterministic scheduling problems belong to the class of COPs the same type of solution methods come into operation. These methods are generally divided into exact and approximate methods (compare Section 5).

Exact Methods The results from complexity theory enable us to distinguish between hard and easy problems (see Section 5.2 for a detailed examination of the concept of $\mathcal{NP}$-completeness). Most scheduling problems belong to the class of $\mathcal{NP}$-hard problems for which it is widely accepted that no optimal method with polynomial run time exists. Since it is considered impossible to optimally solve $\mathcal{NP}$-hard scheduling problems of practical size in a reasonable time, research with practical background focuses on approximate methods that can lead to feasible schedules with an acceptable quality in considerable less time. The reason why problems are hard to solve can be found in their complexity, their size, their specific structure, or a combination of these aspects; cf. (Talbi, 2009).

The most obvious idea to solve a COP is to just enumerate all feasible solutions. But due to the complexity of most COPs, a simple complete enumeration will result in unacceptable high computing times. The challenge is to develop efficient algorithms that perform better than a simple enumeration (Lee, 2004).

The class of exact methods to solve COPs covers Mixed Integer Programming (MIP), Dynamic Programming (DP), Constraint Programming (CP), and Branch and Bound (B&B). Commercial software packages for mathematical programming (e.g. CPLEX[3], Gurobi[4], Xpress[5]) have seen tremendous progress over the last decade in terms of capabilities for solving much larger problem sizes (Méndez et al., 2006). However, for a considerable amount of optimization problems present in academia and industry, it is intractable to obtain optimal solutions by the use of any exact method in a reasonable time. The crucial point is that exact methods need large amounts of time to optimally solve $\mathcal{NP}$-hard problems of practical size. Consequently, the use of exact methods becomes inapplicable in practice, where a responsible person has to make a decision as soon as possible in order to achieve desirable results (Marti and Reinelt, 2011).

Many authors share the opinion that exact methods do not seem to be the method of choice in real-world scheduling systems. Talbi (2009) notes that partial enumerative algorithms such as B&B are limited to rather small instances and thus are not advisable to solve medium and large instances. Potts and Strusevich (2009) notice a stagnation of research on B&B algorithms (and other enumerative approaches), identifying the combinatorial growth of the solution space as an obstacle to the exact solution of practical problems. Although Mönch et al. (2011a) consider MIP and CP as important solution techniques, they argue that these techniques are assessed to be too slow to be adopted in real-world implementations. Among others Dorigo and Stützle (2004), propose approximate methods that trade optimality for efficiency, since the performance of exact algorithms is not satisfactory and their applicability is often limited to rather small instances.

Approximate Methods In contrast to exact methods, approximate methods do not proof the optimality of the obtained solutions; and by definition, this is what differentiates them from exact methods. By not having the burden to proof optimality, approximate methods leave parts of the state space unvisited and thus lead to near-optimal solutions in a reasonable time compared to exact algorithms (Dorigo and Stützle, 2004; Talbi, 2009).

Especially for $\mathcal{NP}$-hard problems, exact algorithms perform poor with respect to computing time. Consequently, solving large instances with exact methods is practical impossible, i.e. would take enormous amounts of time to obtain the optimal solution. Approximate methods trade optimality for efficiency (Dorigo and Stützle, 2004); cf. (Talbi, 2009).

[3] http://www.ibm.com
[4] http://www.gurobi.com
[5] http://www.fico.com

Approximate methods can be further divided into heuristics and approximation algorithms (compare Section 5). A heuristic is any approach without a formal guarantee of performance. Approximation algorithms guarantee that the obtained solution lies within a defined range of the global optimum (Brucker, 2007; Talbi, 2009). An approximation algorithm for a certain problem class guarantees that any obtained solution corresponds to an objective value for which a factor defines the distance to the actual optimum (compare Section 6.4). Basically, Polynomial Time Approximation Schemes (PTASs) and a Fully Polynomial Time Approximation Schemes (FPTASs) form this group (Brucker, 2007; Chandru and Rao, 2010; Klein and Young, 2010).

Within the class of heuristics, it is basically distinguished between constructive heuristics and search heuristics (Zäpfel et al., 2010); cf. (Talbi, 2009). Construction algorithms describe an incremental procedure: starting from an empty initial solution, construction algorithms iteratively add solution components until a complete solution is obtained without any backtracking. Constructive heuristics are usually problem-specific, non-iterative, and create one single solution by applying a set of rules based on problem-specific knowledge. This is the underlying concept of a typical dispatching system. Search heuristics follow a certain search scheme that repeatedly examines many different solutions for a given problem in order to find better solutions (Zäpfel et al., 2010); cf. (Talbi, 2009).

Metaheuristics Search heuristics correspond to metaheuristics in a broader sense. Informally, a metaheuristic states an algorithmic advancement of a simple heuristic, which is commonly defined as a rule of thumb that leads to near-optimal solutions without complete knowledge of the problem. Even though there is no universal definition for metaheuristics, it is widely accepted that a metaheuristic is a general algorithmic framework that *a*) is generally problem-independent and applicable to a wide set of different problems, *b*) describes an iterative upper-level strategy that guides the operations of subordinate heuristics, *c*) combines different concepts for exploring and exploiting the search space (diversification and intensification), often facilitated by the use of randomness (Blum and Roli, 2003; Zäpfel et al., 2010).

In contrast to exact methods, metaheuristics lead to acceptable solutions in a reasonable time; solution quality and computing time is generally not exactly defined, e.g. acceptable and reasonable (Talbi, 2009). Metaheuristics primarily justify their use with a well-balanced performance characteristic that describes a favorable trade-off between solution quality and computing time.

Most authors consistently distinguish between two classes of metaheuristics: trajectory methods based on a single solution and population-based methods (Blum and Roli, 2003); cf. (Talbi, 2009; Luke, 2009; Zäpfel et al., 2010; Marti and Reinelt, 2011; Baghel et al., 2012). Trajectory methods (see Section 5.5) obtain improved solutions by repeatedly modifying an existing solution during the search procedure, e.g. Local Search (LS) (Hill-Climbing), Simulated Annealing (SA), Threshold Accepting (TA), Tabu Search (TS), Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS), Guided Local Search (GLS), and Iterated Local Search (ILS); cf. (Blum and Roli, 2003; Talbi, 2009; Baghel et al., 2012; Boussaïd et al., 2013). Population-based methods (see Section 5.6), operating on a set of solutions, create improved solutions by recombining existing solutions e.g. Evolutionary Algorithms (EAs) (Genetic Algorithms (GAs), EAs, Evolutionary Programming (EP), Genetic Programming (GP)), Scatter Search (SS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Artificial Immune System (AIS); cf. (Blum and Roli, 2003; Talbi, 2009).

Beyond trajectory methods and population-based methods, research spends a considerable effort in developing hybrid metaheuristics, which refer to the idea of combining metaheuristics with metaheuristics or other techniques for optimization. Hybridization aims to exploit the complementary character of different optimization strategies. In fact, combining an appropriate set of complementary algorithmic concepts can be the key for the design of high-performing search methods (Blum et al., 2011). Another research direction with the goal to improve search efficiency deals with parallelization. The idea of parallelizing metaheuristics is driven by two forces: the complexity of computational problems and the rapid development in the technology of distributed computing systems. Here, parallelism can help to reduce the computation time and the increase solution quality (Alba, 2005).

Metaheuristics versus Mixed Integer Programming (MIP) Metaheuristics and LS methods in particular are often the method of choice for real-life scheduling problems with a variety of complicating constraints, since these algorithms can obtain good quality solutions within a reasonable time (Méndez et al., 2006; Potts and Strusevich, 2009).

Several authors document the superiority of metaheuristics compared to MIP approaches in experimental studies, in particular for batch scheduling problems. Xu et al. (2012) show that ACO is more robust and consistently outperforms MIP, especially for large job instances. Melouk et al. (2004), Chang et al. (2004) and Damodaran et al. (2007) describe experiments in which SA consistently outperforms MIP with respect to solution quality and run time. Xu and Bean (2007), Damodaran et al. (2006) and Chou and Wang (2008) provide experimental results indicating that solution schemes based on GAs result in better objective values in shorter run times, especially for larger problems. The experiments in (Klemmt et al., 2009) turn out that VNS outperforms MIP with respect to solution quality and time; They find out that VNS performs faster than MIP while providing high quality solutions at the same time.

Klemmt et al. (2011) show that MIP approaches are basically suitable for optimizing small or medium batch scheduling problems, but combined with decomposition methods. However, even the finding of a feasible solution can be a problem for bigger problem instances (Klemmt et al., 2008). The commercial solver CPLEX is most often used as state-of-the-art solver for MIP formulations. In awareness of the observation that even state-of-the-art MIP solver can only handle relatively small problems, Klemmt et al. (2009) propose decomposition approaches in order to reduce the problem complexity. Although Mönch et al. (2011a) consider MIP and CP as important solution techniques, they argue that these techniques are assessed to be too slow to be adopted in real-world implementations, compared to the vast academic literature about scheduling in semiconductor manufacturing.

Variable Neighbourhood Search (VNS) From the No-Free-Lunch Theorems (NFLTs) it can be informally deduced that no metaheuristic performs better than another across all possible problems (Wolpert and Macready, 1997). Ho and Pepyne (2001) point out that specializing search algorithms to the landscape structure of the focused problem class is the only way one strategy can outperform another (see Section 5.3). Thus, it can be said that the choice of a particular metaheuristic is less important than its actual implementation.

Since performance is a priori no decisive criterion for choosing a metaheuristic method, the aspect of simplicity becomes more important. Simpler algorithms are easier to implement, maintain, adapt, explain and analyze. This statement in general holds for the development of software systems, and in particular for the design of metaheuristic algorithms. Indeed, simplicity results in lower susceptibility to errors (Silberholz and Golden, 2009); cf. (Talbi, 2009).

VNS was first mentioned in (Mladenović and Hansen, 1997) and then examined in different variants in (Hansen and Mladenović, 2001, 2003; Hansen et al., 2001, 2009). The basic idea is a systematic change of the neighborhood during the search, typically established in two alternating search phases, a descent search phase and a randomized perturbation phase. VNS is a relatively new method among metaheuristics, but has attracted many researches to adopt it as solution method for combinatorial optimization problems, e.g. for solving the Traveling Salesman Problem (TSP) (Hu and Raidl, 2008; da Silva and Urrutia, 2010).

But in particular, VNS is used to solve scheduling problems. Implementations of VNS for batch scheduling problems can be found in (Klemmt et al., 2009; Almeder and Mönch, 2011; Kohn and Rose, 2012; Cakici et al., 2013; Kohn and Rose, 2013; Kohn et al., 2013). Single machine scheduling problems are solved by VNS in (Wang and Tang, 2009; Kirlik and Oguz, 2012). VNS for parallel machine scheduling problems is proposed in (Anghinolfi and Paolucci, 2007; de Paula et al., 2007; Driessel and Mönch, 2009; Behnamian et al., 2009; Sevkli and Uysal, 2009; Driessel and Mönch, 2011; Bilyk and Mönch, 2012; Chen and Li Jun-qing, 2012). But also job shop problems are subject to VNS approaches; cf. (Sevkli and Aydin, 2006; Pongchairrerks and Kachitvichyanukul, 2007; Sevkli and Aydin, 2007; Aydin and Sevkli, 2008; Jie et al., 2008; Roshanaei et al., 2009; Adibi et al., 2010; Jun-Qing et al., 2010; Yazdani et al., 2010).

1.4 Goals and Structure of the Thesis

The main goal of this work is to implement a scheduling framework to be deployed as an operational batch scheduling system in the diffusion and oxidation area of a waferfab frontend. The desired scheduling framework additionally requires to offer the opportunity to deal with academic questions beyond its intended use as a prototype in industry. Summarizing, the framework's design needs to enable two main purposes: *a*) operational scheduling on the shop floor (use as a prototype to be deployed in the industry) and *b*) exhaustive studies in an experimental system (use as an experimental system answering methodical questions).

On one hand the intention of the framework's design is to provide a functioning prototype that is suitable to run as a real-time scheduling system on the operational level. The goal is to report valuable practical experiences from implementing a scheduling system into a real-world industrial environment. On the other hand the framework needs to cover an experimental system that offers the capability to properly investigate academic questions in the area of metaheuristic batch scheduling. The goal here is to employ extensive experiments in order to closely examine a particular metaheuristic (Variable Neighborhood Search) solving typical batch scheduling problems in numerous variants.

Furthermore, this thesis claims to cover both aspects from theory and practice in nearly equal measures. The theoretical part intends to facilitate the development of the framework by providing the underlying (theoretical) background in more detail. The focus here lies on two topics. *a*) First this work provides an extensive review about the current state-of-the-art in the area of batch scheduling research. *b*) Second the theoretical groundwork of this work comprises a detailed analysis of the complexity status of the most common batch scheduling problems.

The Prototype The main intention of the framework's design is to provide a functioning prototype that is suitable to serve as a real-time scheduling system on the operational level.

Both researchers and practitioners face the problem that the scheduling problems in academia and industry are rarely identical. The practical problems from industry are typically afflicted with many different constraints. Hence it is often necessary to design case-specific solutions beyond those reported as standard in the scientific literature. Another challenge is to integrate sophisticated scheduling solutions with the existing software structure. The development of real-time simulation/optimization systems as well as their plug-and-play interoperability with existing software is a grand challenge today (Fowler and Rose, 2004).

The main goal of the practical part is to design, implement, and test a proper scheduling system in order to deploy it in a waferfab that relies on dispatching to that date. The resulting scheduling framework requires to comprises two main components: *a*) a top-level scheduling system with all its modeling, simulation and optimization functionalities and *b*) an underlying data level connected to the waferfab's Manufacturing Execution System (MES).

The Experimental System The second important intention of the framework's design besides a functioning prototype is to provide the capability of running massive amounts of experiments in a comfortable fashion. The intended framework is required to offer the capabilities to analyze the effect of the framework's system factors on the system performance in a reproducible environment. Designing, defining, executing and analyzing experiments as time-saving as possible is one of the central functionalities of this framework.

The motivation behind this is that very little is known about the functioning of metaheuristics. The reasons why metaheuristics work so well (and under what conditions) remain unidentified to a large extend. The goal is to identify important factors affecting optimization results by use of intense experimentation. For example, beyond the nature of the search scheme, it is known that a search method's performance also depends on the fine tuning of the algorithm's control parameters (Watson, 2009).

Another tasks of the desired experimental system is to determine the benefit which could be expected from deploying a scheduling system in the real world. The experiments serve to provide reliable numbers for economic benefits based on which a manager decides whether installing a scheduling system is profitable.

A question with more scientific background is that of the sources of the observed improvements considered as optimization effects. The desired framework is required to facilitate answering the question whether the problem instance's characteristics or the scheduling method settings have a greater impact on the improvements. The most important results expected to be the result of the experiments fall in two categories: *a*) insights related to model characteristics and *b*) insights related to method settings.

1.4.1 Structure of the Thesis

The theoretical groundwork facilitates the development of the framework and rounds out the thesis with its practical focus on developing an experimental system and a prototype system. The focus of the theoretical part lies on two topics. *a*) The current state-of-the-art in the area of batch scheduling research (Section 2). *b*) A detailed analysis of the complexity status of the most common batch scheduling problems (Section 6). A well structured system of sections leads topic-by-topic to the main results of this work.

Figure 3 depicts the structure of the thesis comprising nine sections without appendices. The visualized structure basically uses an analogy of a house with three floors: *a*) the fundamentals (Section 1 and Section 2), *b*) the basic topics (Section 3 to Section 5), and *c*) the main topic (Section 6 to Section 8). It ends up with the conclusions in Section 9, i.e. with the roof.

Section 1 and Section 2 provide the fundamentals of this work. The introduction in Section 1 clarifies the motivation, introduces the problem statement, and discusses the related methodology. Section 2 provides an extensive literature review about batch scheduling in wafer fabrication as one of the main pillars of this work. The review covers 170 publications in total, methodically structured in 16 groups.

The second level introduces three basic topics required to fully understand the main part. It begins with a brief overview of wafer fabrication and its most important aspects in Section 3. Section 4 introduces the art of simulation as a key technique in modern manufacturing systems. The theoretical groundwork is complemented with a short overview on the state-of-the-art in metaheuristic optimization (Section 5).

The third level begins with theoretically examining the main topic: batch scheduling. Section 6 provides a detailed introduction to the batch scheduling topic with an detailed analysis of the complexity results of the most common batch scheduling problems. Section 7 gives a detailed description of the implemented framework, providing a top level description of data systems, data transfer mechanisms and essential data procedures. Section 8 reports the insights resulting from the experiments.

Finally, Section 9 comprises the main results of this work, organized in three main topic areas: *a*) theoretical background and state-of-the-art (Section 9.1), *b*) valuable insights spawned by experiments (Section 9.2), and *c*) experiences from implementing a prototype (Section 9.3).

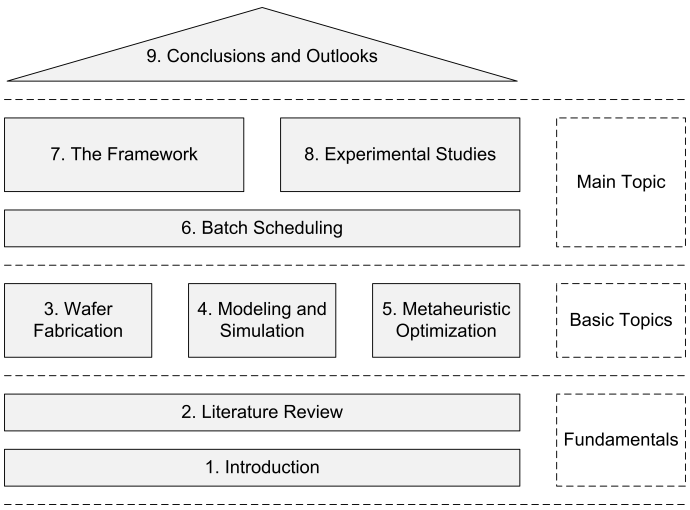
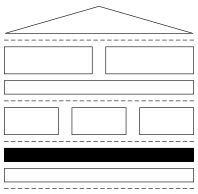


Figure 3: Structure of the thesis



2 Literature Review

Contents

This section provides an extensive review of bounded parallel batch ($p-batch$) scheduling problems, addressed for both single and parallel machine environments under varying sets of constraints. Three constraints are identified as the most important and distinctive characteristics among the vast number of literature related to $p-batch$ scheduling problems: *a*) incompatible job families ($fmls$), *b*) release dates (r_j), and *c*) non-identical job sizes (B, s_j).

The review covers 170 publications in total, methodically structured in 16 groups representing different combinations of $fmls$, r_j and B, s_j for the single and the parallel machine environment; cf. Table 1. For an overview on the entire set of literature sources see the tables in Appendix A. A clear emphasis is put on deterministic scheduling problems. However, for the sake of completeness stochastic scheduling models gain also recognition in the context of real-time control as well. The reviewed works most often relate to $p-batch$ scheduling problems found in waferfabs, whereas minor publications are motivated by other industrial applications. The majority is either motivated by frontend diffusion/oxidation operations or by backend burn-in operations.

Prior some authors have presented literature reviews about batch scheduling problems. To our best knowledge, Potts and van Wassenhove (1992) and Webster and Baker (1995) provide the first reviews about batching problems, discussing early research in the field of batch scheduling theory with focus on single-machine scheduling models. Potts and Kovalyov (2000) give an extended review of batching problems in various machine environments and for several objectives, putting emphasis on complexity results as well as on the efficiency and effectiveness of algorithms. Mathirajan and Sivakumar (2003) especially review batch scheduling problems in semiconductor manufacturing, classifying batching problems into 12 groups while distinguishing between stochastic and deterministic problems. They refine their classification schemes and systematically organize the published articles in an updated survey three years later (Mathirajan and Sivakumar, 2006b).

Table 1: Methodical literature review in 16 groups

machine environment	constraints B, s_j	r_j	$fmls$	$\alpha \mid \beta \mid \gamma$-notation	section	#publications
$1 \mid \cdot \mid \cdot$	-	-	-	$1 \mid p-batch, b<n \mid \cdot$	2.1.1	21
	✓	-	-	$1 \mid p-batch, B, s_j \mid \cdot$	2.1.2	24
	-	✓	-	$1 \mid p-batch, b<n, r_j \mid \cdot$	2.1.3	20
	-	-	✓	$1 \mid p-batch, b<n, fmls \mid \cdot$	2.1.4	17
	✓	✓	-	$1 \mid p-batch, B, s_j, r_j \mid \cdot$	2.1.5	11
	✓	-	✓	$1 \mid p-batch, B, s_j, fmls \mid \cdot$	2.1.6	7
	-	✓	✓	$1 \mid p-batch, b<n, r_j, fmls \mid \cdot$	2.1.7	19
	✓	✓	✓	$1 \mid p-batch, B, s_j, r_j, fmls \mid \cdot$	2.1.8	2
$Pm \mid \cdot \mid \cdot$	-	-	-	$Pm \mid p-batch, b<n \mid \cdot$	2.2.1	2
	✓	-	-	$Pm \mid p-batch, B, s_j \mid \cdot$	2.2.2	10
	-	✓	-	$Pm \mid p-batch, b<n, r_j \mid \cdot$	2.2.3	5
	-	-	✓	$Pm \mid p-batch, b<n, fmls \mid \cdot$	2.2.4	7
	✓	✓	-	$Pm \mid p-batch, B, s_j, r_j \mid \cdot$	2.2.5	10
	✓	-	✓	$Pm \mid p-batch, B, s_j, fmls \mid \cdot$	2.2.6	2
	-	✓	✓	$Pm \mid p-batch, b<n, r_j, fmls \mid \cdot$	2.2.7	22
	✓	✓	✓	$Pm \mid p-batch, B, s_j, r_j, fmls \mid \cdot$	2.2.8	7
						170 (total)

Machine Environment This review examines single machine models in Section 2.1 and parallel machines models in Section 2.2. The grouping comprises eight variants of the basic single machine problem $1 \mid p-batch, b < n \mid \cdot$ and eight variants of the basic parallel machines problem $Pm \mid p-batch, b < n \mid \cdot$; cf. Table 1.

Beyond single and parallel machine models and out of scope of this review, the area of scheduling problems in flowshop environments with BPMs attracts a remarkable amount of researchers. Among others, Ahmadi et al. (1992), Sung and Yoon (1997), Sung et al. (2000), Sung and Min (2001) and Damodaran and Srihari (2004) deal with deterministic batch scheduling problems in flowshops. In contrast, among others Gurnani et al. (1991, 1992), Neale and Duenyas (2000), van der Zee (2002), Mason et al. (2007), and Ham and Fowler (2008) study batch scheduling problems in flowshops under stochastic settings.

Primary Constraints Three primary constraints are used to establish a methodical grouping with 16 classes for the literature: *a*) non-identical job sizes (B, s_j), *b*) release dates (r_j), and *c*) incompatible job families $(fmls)$. Refer to Table 1.

The combination of these three constraints result in eight model variants for each machine model, i.e. eight variants for the single machine model and eight variants for the parallel machine model. For example, the eight single machine models range from the basic model with no constraints $(1 \mid p-batch, b < n \mid \cdot)$ to the model with three constraints $(1 \mid p-batch, B, s_j, r_j, fmls \mid \cdot)$. In this way, there exist another eight corresponding parallel machine models, adding up to 16 model variants in total.

From another point of view, there exist eight different models for each of these three characterizing constraints $(B, s_j,\ r_j$ and $fmls)$, whereby the single machine model and the parallel machine model count four variants each. There are eight models that deal with non-identical job sizes (B, s_j); the simplest case $1 \mid p-batch, B, s_j \mid \cdot$ (see Section 2.1.2) is first studied by Uzsoy (1994). The grouping counts eight models that deal with release dates (r_j). The simplest case $1 \mid p-batch, b < n, r_j \mid \cdot$ (see Section 2.1.3) wis first studied by Ikura and Gimple (1986), Lee et al. (1992), and Webster and Baker (1995); cf. Brucker (2007). Consequently there exist eight models that deal with incompatible job families $(fmls)$. The simplest case $1 \mid p-batch, b < n, fmls \mid \cdot$ (see Section 2.1.4) is first studied by Chandru et al. (1993b) and Uzsoy (1995), whereby Uzsoy (1995) also examines the case with job arrivals.

Process Time Models There exist two basic model types: *a*) the longest job processing time model (L) and *b*) the family processing time model (F).

The longest job processing time model (L) describes the case in which the jobs have arbitrary processing times and the processing time of the batch is determined by the longest processing time of its jobs. This kind of model is originally motivated by the backend burn-in operations. The simplest case $1 \mid p-batch, b < n \mid \cdot$ is first studied by Lee et al. (1992), Albers and Brucker (1993) and Chandru et al. (1993a); cf. Brucker (2007).

The family processing time model (F) is usually motivated by diffusion or oxidation operations that deal with incompatible job families. In this case, the processing time of a batch is given by the corresponding job family and different job families have different processing times. As one of the first, Chandru et al. (1993b) and Uzsoy (1995) study the simplest case $1 \mid p-batch, b < n, fmls \mid \cdot$, whereby Uzsoy (1995) considers the case with job arrivals $1 \mid p-batch, b < n, r_j, fmls \mid \cdot$. Fowler et al. (1992a,b) and Weng and Leachman (1993) propose real-time control strategies that incorporate future arrivals for the same problem under stochastic settings.

In general $p-batch$-scheduling problems formally characterized with incompatible job families $(fmls)$ correspond to the family processing time model (F). Minor works consider incompatible job families $(fmls)$ in conjunction with the longest processing time model (L); cf. (Boudhar, 2003; Finke et al., 2008; Nong et al., 2008a; Sabouni and Jolai, 2010; Meng and Lu, 2011).

Some authors consider the special case with constant processing times (C), i.e. the processing time is generally a constant that is identical for all jobs. For example, Webster and Baker (1995), Ozturk et al. (2012), Li et al. (2012b), and Koehler and Khuller (2013) deal with deterministic models characterized by constant processing times, whereas Ikura and Gimple (1986), Glassey and Weng (1991), Fowler et al. (1992a), van der Zee et al. (1997), Korkmaz (2004) and van der Zee (2007) discuss real-time-control strategies in stochastic environments.

Methods and Objectives The focus of this review lies on deterministic offline scheduling models. The solution approaches are basically distinguished into *a*) exact methods (E), *b*) heuristics (H), *c*) metaheuristic (MH), and *d*) approximation algorithms (A); exact methods for special case(s) are denoted with the character (*).

Despite of the fact that the emphasis is put on deterministic offline scheduling models, the review also mentions deterministic scheduling models under online setting (#) and stochastic scheduling models related to real-time control. For example, Chen et al. (2001), Nong et al. (2008b), Meng and Lu (2011), Zhang et al. (2001a) and Li et al. (2012b,a) consider online-scheduling models. The most important strategies in the area of real-time control (RTC) are discussed in Section 6.8 more in detail.

This review aims to describe the objective function as accurately as possible, e.g. the objective is to minimize makespan (C_{max}). At least, the type of objective is determined, i.e. related to cycle time (CT), related to on-time delivery (OTD) or composed of multiple objectives (MO).

Additional Constraints Beyond the three primary constraints (B, s_j, r_j, $fmls$), there exist numerous constraints considered in conjunction with $p-batch$ scheduling problems. This review lists models with many different extensions, e.g. deadlines ($\bar{d}_j$), machine eligibility constraints (M_j), sequence-dependent setup times ($sdst$), preemption ($prmpt$), rejection ($rjct$), reentrant jobs ($rntr$), job splitting ($jspl$), graph compatibility (gc), secondary resources (sr), stochastic processing times (spt), precedence constraints ($prec$), and machine breakdowns ($brkdwn$).

2.1 Single Machine Batch Scheduling Problems

Despite of the fact that scheduling a single machine is quite unattractive from practical point of view, research is very interested in single machine models since they form sub problems for parallel machine scheduling problems typically faced in industry; cf. (Mönch et al., 2011a). This section covers eight variants of single BPM scheduling problems summarized in Table 2.

Table 2: Eight single machine scheduling problems

machine environment	constraints B, s_j	r_j	$fmls$	$\alpha \mid \beta \mid \gamma$-notation	section
$1 \mid \cdot \mid \cdot$	-	-	-	$1 \mid p-batch, b<n \mid \cdot$	2.1.1
	✓	-	-	$1 \mid p-batch, B, s_j \mid \cdot$	2.1.2
	-	✓	-	$1 \mid p-batch, b<n, r_j \mid \cdot$	2.1.3
	-	-	✓	$1 \mid p-batch, b<n, fmls \mid \cdot$	2.1.4
	✓	✓	-	$1 \mid p-batch, B, s_j, r_j \mid \cdot$	2.1.5
	✓	-	✓	$1 \mid p-batch, B, s_j, fmls \mid \cdot$	2.1.6
	-	✓	✓	$1 \mid p-batch, b<n, r_j, fmls \mid \cdot$	2.1.7
	✓	✓	✓	$1 \mid p-batch, B, s_j, r_j, fmls \mid \cdot$	2.1.8

2.1.1 1 | p-batch, b < n | ·

The review begins with the basic bounded batch scheduling problem on a single BPM, denoted with $1 \mid p-batch, b<n \mid \cdot$. All publications describe the longest job processing time model (L), with the exception of the model in (Webster and Baker, 1995) that deals with constant processing times (C). See Table 3 for a tabular overview.

Makespan Minimizing C_{max} on a single BPM can be solved optimally in polynomial time. Brucker et al. (1998) and Sung and Choung (2000) present such algorithms with polynomial run time; cf. Brucker (2007).

On-Time Delivery Lee et al. (1992) present polynomial algorithms for special cases of L_{max} and $\sum U_j$; cf. (Webster and Baker, 1995). Brucker and Kovalyov (1996) and Liu (2007) provide polynomial and pseudo-polynomial DP approaches for special cases of $\sum w_j U_j$; cf. Brucker (2007).

Brucker and Kovalyov (1996) also provide a FPTAS for the original, unrestricted problem $\sum w_j U_j$. The problem $\sum T_j$ can be solved polynomially if the processing times are constant (Brucker, 2007) and pseudo-polynomially if the processing times and due dates are agreeable (Liu, 2007). Mönch et al. (2006a) incorporate deadlines and describe a GA with the objective of minimizing the sum of the absolute deviations of completion times from the due date, assuming that the jobs have the same due date.

Cycle Time Webster and Baker (1995), Hochbaum and Landy (1997), Brucker et al. (1998) and Poon and Yu (2004) provide polynomial and pseudo-polynomial algorithms for special cases of $\sum C_j$. For the unrestricted version, Chandru et al. (1993a) give a B&B algorithm, Hochbaum and Landy (1997) describe an approximation algorithm, and Cai et al. (2002) and Deng et al. (2002) describe PTASs. Chandru et al. (1993a) present a simple heuristic to solve the same problem. Turning to the weighted variant, Albers and Brucker (1993), Webster and Baker (1995) and Brucker (2007) present polynomial algorithms for special cases of $\sum w_j C_j$ and Uzsoy and Yang (1997) give a B&B scheme as well as several heuristics for the original, unrestricted version. Chen et al. (2001) study the same problem under online setting.

Multiple Objectives Sabouni and Jolai (2010) consider a special case of $1 \mid p-batch, b<n \mid \cdot$ with jobs that belong to two different customers and simultaneously optimize for C_{max} and L_{max}. In this case, the jobs belong to different customers, processed based on their individual criteria, i.e. C_{max} or L_{max}. They optimally solve the problem with equal processing times.

Real-Time Control Neuts (1967) study the problem under stochastic settings and present the well-known Minimum Batch Size (MBS) rule that aims on minimizing CT. Ganesan et al. (2004) study the problem $1 \mid p-batch, b<n \mid \cdot$ under stochastic setting and minimize the mean CT and the maximum tardiness at the same time.

Table 3: Publications related to $1 \mid \text{p-batch}, \text{b} < \text{n} \mid \cdot$

publication	model	method	objective	constraints
Neuts (1967)	-	RTC	CT	-
Lee et al. (1992)	L	E*	OTD	-
Albers and Brucker (1993)	L	E*	CT	-
Chandru et al. (1993a)	L	E, H	CT	-
Webster and Baker (1995)	C	E*	OTD, CT	-
Brucker and Kovalyov (1996)	L	E*, A	OTD	-
Hochbaum and Landy (1997)	L	E*, A	CT	-
Uzsoy and Yang (1997)	L	E, H	CT	-
Brucker et al. (1998)	L	E*	CT	-
Brucker et al. (1998)	L	E	C_{max}	-
Sung and Choung (2000)	L	E	C_{max}	-
Chen et al. (2001)	L	A#	CT	-
Cai et al. (2002)	L	A	CT	-
Deng et al. (2002)	L	A	CT	-
Ganesan et al. (2004)	L	RTC	MO	-
Poon and Yu (2004)	L	E	CT	-
Mönch et al. (2006a)	L	MH*	OTD	$\bar{d}_j$
Brucker (2007)	L	E	C_{max}	-
Brucker (2007)	L	E*	OTD, CT	-
Liu (2007)	L	E*	OTD	-
Sabouni and Jolai (2010)	L	E*	MO	-

model:{longest job processing time (L), constant processing time (C)}; **method:**{exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*), online setting (#)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{deadlines ($\bar{d}_j$)}

2.1.2 1 | p-batch, B, s_j | ·

This section reviews the basic batch scheduling problem on a single BPM with non-identical job sizes, denoted with $1 \mid p-batch, B, s_j \mid \cdot$. The mentioned publications deal with the longest job processing time model (L), with the exception of the model in (van der Zee, 2007) that deals with constant processing times (C). See Table 4 for a tabular overview.

Makespan Dupont and Dhaenens-Flipo (2002) and Parsa et al. (2010) propose B&B methods to find solutions with optimal C_{max}. Zhang et al. (2001b), Kashan et al. (2009) and Zhang and Cao (2007) present approximation algorithms and Uzsoy (1994) and Parsa et al. (2010) provide heuristics. Due to the complexity of the problem, several authors developed metaheuristics for it. Mathirajan et al. (2004), Melouk et al. (2004) and Damodaran et al. (2007) present SA algorithms. Damodaran et al. (2006) and Kashan et al. (2006a,b) propose GAs. Jia and Leung (2014), Cheng et al. (2010) and Zhang et al. (2009b) give descriptions of methods based on the ACO concept, whereby Cheng et al. (2010) consider stochastic processing times.

On-Time Delivery Malapert et al. (2012) present a CP approach for L_{max} and Erramilli and Mason (2008) minimize $\sum w_j T_j$ with MIP and SA.

Cycle Time Uzsoy (1994) presents a B&B scheme and heuristics for $\sum C_j$. Jolai Ghazvini and Dupont (1998) also describe a heuristic and Xu et al. (2008a,b) propose an ACO approach for the same problem. Azizoglu and Webster (2000) propose a B&B scheme for $\sum w_j C_j$.

Multiple Objectives Kashan et al. (2010) present different GAs for minimizing C_{max} and L_{max} at the same time. Lu et al. (2009a) present a PSO algorithm that simultaneously minimizes C_{max} and $\sum C_j$.

Table 4: Publications related to $1 \mid \text{p-batch}, \text{B}, \text{s}_\text{j} \mid \cdot$

publication	model	method	objective	constraints
Uzsoy (1994)	L	H	C_{max}	-
Uzsoy (1994)	L	E, H	CT	-
Jolai Ghazvini and Dupont (1998)	L	H	CT	-
Azizoglu and Webster (2000)	L	E	CT	-
Zhang et al. (2001b)	L	A	C_{max}	-
Dupont and Dhaenens-Flipo (2002)	L	E	C_{max}	-
Mathirajan et al. (2004)	L	MH	C_{max}	-
Melouk et al. (2004)	L	E, MH	C_{max}	-
Damodaran et al. (2006, 2007)	L	MH	C_{max}	-
Kashan et al. (2006a,b)	L	MH	C_{max}	-
Zhang and Cao (2007)	L	A	C_{max}	-
Erramilli and Mason (2008)	L	E, MH	OTD	-
Xu et al. (2008a,b)	L	MH	CT	-
Kashan et al. (2009)	L	A	C_{max}	-
Lu et al. (2009a)	L	MH	MO	-
Zhang et al. (2009b)	L	MH	C_{max}	-
Cheng et al. (2010)	L	MH	C_{max}	*spt*
Kashan et al. (2010)	L	MH	MO	-
Parsa et al. (2010)	L	E, H	C_{max}	-
Malapert et al. (2012)	L	E	OTD	-
Jia and Leung (2014)	L	MH	C_{max}	-

model:{longest job processing time (L)}; **method:**{exact method (E), heuristic (H), metaheuristic (MH), approximation algorithm (A)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{stochastic processing times (*spt*)}

2.1.3 1 | p-batch, b < n, r_j | ·

The basic bounded batch scheduling problem on a single BPM with release dates is denoted with $1 \mid p-batch, b < n, r_j \mid \cdot$. The publications describe the longest job processing time model (L), with a few exceptions. Gupta et al. (2004) deal with the family processing time model (F) and Webster and Baker (1995) discuss both the longest job processing time model (L) and the family processing time model (F). Ikura and Gimple (1986) and Glassey and Weng (1991) restrict themselves to constant processing times (C). See Table 5 for a tabular overview.

Makespan Ikura and Gimple (1986) and Sung et al. (2002) present polynomial algorithms for special cases of C_{max}. Ikura and Gimple (1986) also study the problem with deadlines. Lee and Uzsoy (1999), Poon and Zhang (2000), Liu and Yu (2000) and Deng et al. (2003) give pseudo-polynomial algorithms for special cases. Sung and Choung (2000) present a B&B scheme and several heuristics for the general problem. Efficient heuristics can also be found in (Lee and Uzsoy, 1999) and (Liu and Yu, 2000), whereas Liu and Yu (2000) describe a heuristic with a defined bound on performance. Poon and Zhang (2000) and Deng et al. (2003) provide PTASs. Zhang et al. (2001a) study problem under online setting. Lu et al. (2009b) and Cao and Yang (2009) incorporate job rejection in their models. Lu et al. (2009b) give polynomial and pseudo-polynomial algorithms for special cases as well as a approximation algorithm and a PTAS. Cao and Yang (2009) also present a PTAS for it.

On-Time Delivery Webster and Baker (1995), Baptiste (2000), Li and Lee (1997) and Lee et al. (1992) show that special cases of L_{max} can be solved in polynomial time. Wang and Uzsoy (2002) present a GA combined with DP to solve the general version. Lee et al. (1992) and Lee and Pinedo (1997) present polynomial algorithms for special cases of $\sum U_j$. Baptiste (2000) and Brucker (2007) consider the weighted variant $\sum w_j U_j$, presenting algorithms with polynomial run time for special cases.

Cycle Time Liu and Cheng (2005) give a PTAS for $\sum C_j$ and Webster and Baker (1995) present a polynomial DP for a special case of the same problem. Baptiste (2000) and Brucker (2007) present polynomial algorithms for special cases of the weighted version $\sum w_j C_j$

Real-Time Control Glassey and Weng (1991) and van der Zee (2004) present look-ahead strategies in a stochastic environment in order to reduce CTs.

2.1.4 1 | p-batch, b < n, fmls | ·

This section reviews the basic bounded batch scheduling problem with incompatible families on a single BPM, denoted with $1 \mid p-batch, b < n, fmls \mid \cdot$. The publications describe the family processing time model (F), with a few exceptions. Boudhar (2003), Finke et al. (2008), Nong et al. (2008b), Sabouni and Jolai (2010) and Meng and Lu (2011) deal with the the longest job processing time model (L). See Table 6 for a tabular overview.

Makespan Uzsoy (1995) shows that minimizing C_{max} is optimally solvable in polynomial time. Nong et al. (2008b) and Meng and Lu (2011) study the problem under online setting and provide approximation algorithms with a worst case ratio 2. Finke et al. (2008) and Boudhar (2003) study the problem with compatibility graphs, whereupon both present polynomial algorithms for special cases. Boudhar (2003) also gives a heuristic.

On-Time Delivery Uzsoy (1995) shows that minimizing L_{max} is optimally solvable in polynomial time. Dauzère-Pérès and Mönch (2013) provide a MIP formulation and a GA for minimizing $\sum U_j$ and the weighted case $\sum w_j U_j$. Jolai (2005) present polynomial and pseudo-polynomial algorithms for special cases of $\sum U_j$ minimization. Liu and Zhang (2008) provide a DP method for minimizing $\sum w_j U_j$. Mehta and Uzsoy (1998) present a polynomial DP scheme for a special case and several heuristics for the general case of $\sum T_j$ minimization. Devpura et al. (2001) and Perez et al. (2005) propose several heuristics to solve the weighted case $\sum w_j T_j$.

Table 5: Publications related to $1 \mid \text{p-batch}, \text{b} < \text{n}, \text{r}_\text{j} \mid \cdot$

publication	model	method	objective	constraints
Ikura and Gimple (1986)	C	E*	C_{max}	d_j
Glassey and Weng (1991)	C	RTC	CT	-
Lee et al. (1992)	L	E*	OTD	-
Webster and Baker (1995)	L, F	E*	OTD, CT	-
Li and Lee (1997)	L	E*	OTD	-
Lee and Uzsoy (1999)	L	E*, H	C_{max}	-
Baptiste (2000)	L	E*	OTD, CT	-
Liu and Yu (2000)	L	E*, A	C_{max}	-
Poon and Zhang (2000)	L	E*, A	C_{max}	-
Sung and Choung (2000)	L	E, H	C_{max}	-
Zhang et al. (2001a)	L	A#*, A#	C_{max}	-
Sung et al. (2002)	L	E*	C_{max}	-
Wang and Uzsoy (2002)	L	MH	OTD	-
Deng et al. (2003)	L	E*, A	C_{max}	-
Gupta et al. (2004)	F	RTC	OTD	-
van der Zee (2004)	L	RTC	CT	-
Liu and Cheng (2005)	L	A	CT	-
Brucker (2007)	L	E*	OTD, CT	-
Cao and Yang (2009)	L	A	MO	$rjct$
Lu et al. (2009b)	L	E*, A	MO	$rjct$

model:{longest job processing time (L), constant processing time (C), family processing time (F)}; **method:**{exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*), online setting (#)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{rejection ($rjct$),deadlines (d_j)}

Cycle Time Chandru et al. (1993b) propose a polynomial DP algorithm for a special case of minimizing $\sum C_j$. Uzsoy (1995) shows that minimizing the weighted case $\sum w_j C_j$ can be solved in polynomial time.

Multiple Objectives Sabouni and Jolai (2010) consider a special case of $1 \mid p-batch, b < n, fmls \mid \cdot$ with jobs that belong to two different customers. The objective function simultaneously optimizes for C_{max} and L_{max}, given that the jobs belong to different customers processed based on their individual criteria, i.e. C_{max} or L_{max}. They optimally solve the problem with equal processing times and give a heuristic for different processing times.

Real-Time Control Duenyas and Neale (1997) deal with random processing times and develop a simple heuristic scheduling policy that minimizes waiting costs. Kim et al. (1998) and Kim et al. (2001) propose batching rules that incorporate downstream and due date information. Akcali et al. (2000) examine the performance of different loading and dispatching policies for batch processing.

2.1.5 $1 \mid \text{p-batch}, \text{B}, \text{s}_\text{j}, \text{r}_\text{j} \mid \cdot$

This section reviews the batch scheduling problem on a single BPM subject non-identical job sizes and release dates, denoted with $1 \mid p-batch, B, s_j, r_j \mid \cdot$. The mentioned publications deal with the longest job processing time model (L), with the exception of the model in (van der Zee, 2007) that deals with constant processing times (C). See Table 7 for a tabular overview.

Makespan MIP formulations for C_{max} can be found in (Xu et al., 2012) and (Vélez-Gallego et al., 2011). Li et al. (2005a) and Lu et al. (2010) present approximation algorithms for it, whereby Lu et al. (2010) allow for job rejections. Zhou et al. (2013) and Vélez-Gallego et al. (2011) and Xu et al. (2012) describe heuristics. Chou et al. (2006) present a GA and Xu et al. (2012) an ACO approach.

On-Time Delivery Mathirajan et al. (2010) give a SA algorithm and Chou and Wang (2008) a GA with integrated DP for $\sum w_j T_j$, and both present MIP formulations and heuristics for it.

Table 6: Publications related to $1 \mid \text{p-batch}, b < n, \text{fmls} \mid \cdot$

publication	model	method	objective	constraints
Chandru et al. (1993b)	F	E*	CT	-
Uzsoy (1995)	F	E	C_{max}, OTD, CT	-
Duenyas and Neale (1997)	F	RTC	CT	*spt*
Kim et al. (1998)	F	RTC	MO	-
Mehta and Uzsoy (1998)	F	E*, H	OTD	-
Akcali et al. (2000)	F	RTC	CT	-
Devpura et al. (2001)	F	H	OTD	-
Kim et al. (2001)	F	RTC	OTD	-
Boudhar (2003)	L	E*, H	C_{max}	*gc*
Jolai (2005)	F	E*	OTD	-
Perez et al. (2005)	F	H	OTD	-
Finke et al. (2008)	L	E*	C_{max}	*gc*
Liu and Zhang (2008)	F	E	OTD	-
Nong et al. (2008b)	L	A#	C_{max}	-
Sabouni and Jolai (2010)	L	E*, H	MO	-
Meng and Lu (2011)	L	A#	C_{max}	-
Dauzère-Pérès and Mönch (2013)	F	E, MH	OTD	-

model:{longest job processing time (L), family processing time (F)}; **method:**{exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*), online setting (#)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{graph compatibility (*gc*), stochastic processing times (*spt*)}

Cycle Time Chang and Wang (2004) present a heuristic algorithm for $\sum C_j$.

Multiple Objectives Wang and Chou (2013) describe an exhausted enumeration approach and a GA to find pareto-optimal solutions with respect to C_{max} and $\sum w_j T_j$.

Real-Time Control van der Zee (2007) proposes a look-ahead strategy that deals with non-identical sizes in order or minimize the average flow time in the long run in a stochastic environment.

Table 7: Publications related to $1 \mid \text{p-batch}, B, s_j, r_j \mid \cdot$

publication	model	method	objective	constraints
Chang and Wang (2004)	L	H	CT	-
Li et al. (2005a)	L	A	C_{max}	-
Chou et al. (2006)	L	MH	C_{max}	-
van der Zee (2007)	C	RTC	CT	-
Chou and Wang (2008)	L	E, H, MH	OTD	-
Lu et al. (2010)	L	A	C_{max}	*rjct*
Mathirajan et al. (2010)	L	E, H, MH	OTD	-
Vélez-Gallego et al. (2011)	L	E, H	C_{max}	-
Xu et al. (2012)	L	E, H, MH	C_{max}	-
Wang and Chou (2013)	L	E, MH	MO	-
Zhou et al. (2013)	L	H	C_{max}	-

model:{longest job processing time (L), constant processing time (C)}; **method:**{exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{rejection (*rjct*)}

2.1.6 $1 \mid \text{p-batch}, B, s_j, \text{fmls} \mid \cdot$

This section reviews the bounded batch scheduling problem on a single BPM with incompatible families and non-identical job sizes, denoted with $1 \mid p-batch, B, s_j, fmls \mid \cdot$. The mentioned publications exclusively deal with the family processing time model (F). See Table 8 for a tabular overview.

Makespan Koh et al. (2005) give a mathematical formulation and propose a number of heuristic algorithms as well as a GA to minimize C_{max}. Kempf et al. (1998) examine heuristics for the problem with secondary resource constraints, giving mathematical formulations for special cases.

On-Time Delivery Hoitomt and Luh (1992) present a heuristic based on relaxation, minimizing $\sum w_j T_j^2$.

Cycle Time Koh et al. (2005) also give a mathematical formulation, heuristic algorithms including a GA to minimize $\sum C_j$ and $\sum w_j C_j$. Kempf et al. (1998) study the problem with secondary resource constraints, providing mathematical formulations for special cases and heuristics for the general case. Azizoglu and Webster (2001) present a B&B for $\sum w_j C_j$. Dobson and Nambimadom (1992, 2001) give a mathematical formulation for the same problem, a polynomial algorithm for a special case and several heuristics for the general problem. Kashan and Karimi (2007) develop an ACO method for $\sum w_j C_j$ minimization.

Table 8: Publications related to $1 \,|\, \text{p-batch}, \text{B}, \text{s}_\text{j}, \text{fmls} \,|\, \cdot$

publication	model	method	objective	constraints
Hoitomt and Luh (1992)	F	H	OTD	-
Kempf et al. (1998)	F	E*, H	C_{max}, CT	*sr*
Azizoglu and Webster (2001)	F	E	CT	-
Dobson and Nambimadom (1992, 2001)	F	E, E*, H	CT	-
Koh et al. (2005)	F	E, H, MH	C_{max}, CT	-
Kashan and Karimi (2007)	F	MH	OTD	-

model:{family processing time (F)}; **method:**{exact method (E), heuristic (H), metaheuristic (MH), special case(s) (*)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD)}; **constraints:**{secondary resources (*sr*)}

2.1.7 $1 \,|\, \text{p-batch}, \text{b} < \text{n}, \text{r}_\text{j}, \text{fmls} \,|\, \cdot$

This section reviews the bounded batch scheduling problem on a single BPM with incompatible families and release dates, denoted with $1 \,|\, p-batch, b < n, r_j, fmls \,|\, \cdot$. The publications describe the family processing time model (F), with a few exceptions. Boudhar (2003) and Nong et al. (2008a) deal with the longest job processing time model (L). Fowler et al. (1992a) and Korkmaz (2004) restrict themselves to constant processing times (C). See Table 9 for a tabular overview.

Makespan Uzsoy (1995) presents an efficient optimal algorithm for minimizing C_{max} and Nong et al. (2008a) develeop a PTAS for it.

On-Time Delivery Uzsoy (1995) describes several heuristics for L_{max}. Jia et al. (2013) additionally consider reentrant jobs, proposing a rolling horizon strategy that minimizes L_{max} or $\sum w_j T_j$. Tangudu and Kurz (2006) present a B&B scheme for solving the $\sum w_j T_j$-problem and Kurz and Mason (2008) propose a heuristic for it. Li and Qiao (2008) and Guo et al. (2010) include sequence-dependent setup times and both present algorithms based on ACO.

Cycle Time Korkmaz (2004) and Yao et al. (2012) present B&B schemes for the $\sum C_j$-problem and Tajan et al. (2011) solve it with DP. Korkmaz (2004) allows job splitting and proposes a heuristic for the case with constant processing times. Jia et al. (2013) additionally consider reentrant jobs, proposing a rolling horizon strategy that minimizes $\sum C_j$.

Real-Time Control Several look-ahead strategies consider job families while minimizing CT measures, e.g. (Fowler et al., 1992a,b), (Weng and Leachman, 1993), (Robinson et al., 1995), (Duenyas and Neale, 1997), and (Tajan et al., 2008, 2011). (Duenyas and Neale, 1997) consider stochastic process times. Similarly Gupta et al. (2004) stochastically examine the problem, but propose a look-ahead batching rule that minimizes an objective function with earliness and tardiness measures.

Table 9: Publications related to $1 \mid \text{p-batch}, b < n, r_j, \text{fmls} \mid \cdot$

publication	model	method	objective	constraints
Fowler et al. (1992a,b)	F	RTC	CT	-
Weng and Leachman (1993)	F	RTC	CT	-
Robinson et al. (1995)	F	RTC	CT	-
Uzsoy (1995)	F	E	C_{max}	-
Uzsoy (1995)	F	H	OTD	-
Duenyas and Neale (1997)	F	RTC	CT	*spt*
Boudhar (2003)	L	E*, H	C_{max}	*gc*
Korkmaz (2004)	C	E, H	CT	*jspl*
Gupta and Sivakumar (2006)	F	RTC	OTD, MO	-
Tangudu and Kurz (2006)	F	E	OTD	-
Kurz and Mason (2008)	F	H	OTD	-
Li and Qiao (2008)	F	MH	OTD	*sdst*
Nong et al. (2008a)	L	A	C_{max}	-
Tajan et al. (2008)	F	RTC	CT	-
Guo et al. (2010)	F	MH	OTD	*sdst*
Tajan et al. (2011)	F	RTC, E	CT	-
Yao et al. (2012)	F	E	CT	-
Jia et al. (2013)	F	H	OTD, CT	*rntr*

model:{longest job processing time (L), constant processing time (C), family processing time (F)}; **method:**{exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{sequence dependent setup times (*sdst*), reentrant jobs (*rntr*), job splitting (*jspl*), graph compatibility (*gc*), stochastic processing times (*spt*)}

2.1.8 $1 \mid \text{p-batch}, B, s_j, r_j, \text{fmls} \mid \cdot$

This section reviews the batch scheduling problem with incompatible families on a single BPM subject to non-identical job sizes and release dates, denoted with $1 \mid p-batch, B, s_j, r_j, fmls \mid \cdot$. Nong et al. (2008a) deal with the longest job processing time model (L), whereas Gokhale and Mathirajan (2011) ususes the family processing time model (F). See Table 10 for a tabular overview.

Makespan Nong et al. (2008a) present an approximation algorithm to minimize C_{max}.

On-Time Delivery Gokhale and Mathirajan (2011) incorporate job splitting, presenting a mathematical model and a few heuristics to minimize $\sum w_j T_j$.

Cycle Time To our best knowledge, no publications exist neither for $1 \mid p-batch, B, s_j, r_j, fmls \mid \sum C_j$, nor for the weighted case $\sum w_j C_j$.

Table 10: Publications related to $1 \mid \text{p-batch}, B, s_j, r_j, \text{fmls} \mid \cdot$

publication	model	method	objective	constraints
Nong et al. (2008a)	L	A	C_{max}	-
Gokhale and Mathirajan (2011)	F	E, H	OTD	*jspl*

model:{longest job processing time (L), family processing time (F)}; **method:**{exact method (E), heuristic (H), approximation algorithm (A)}; **objectives:**{makespan (C_{max}), on-time delivery (OTD)}; **constraints:**{job splitting (*jspl*)}

2.2 Parallel Machines Batch Scheduling Problems

A waferfab is typically organized in a number of work areas, given that each work area covers a set of similar machines. In this context, similarity roughly refers to the architecture of the machines as well as to the type of processes they provide. For the entire machine pool in the fab, there exists a number of subsets of machines so that each subset provides a disjunctive set of processes, also referred to as CMSs or work centers. For a typical waferfab such a CMS roughly counts up to up

to several dozens of machines that form a manufacturing entity from the viewpoint of production logistics. Based on the shop floor layout, the machines of a single work center are often located in near proximity, but may be also locally distributed on the shop floor.

The concept of parallel machines describes a manufacturing system consisting of a number of machines in parallel where each job requires a single operation on any of the parallel machines, or on a subset of them in the case of machine eligibility restrictions. For the sake of simplicity, this review does not explicitly distinguish between the three common types of parallel machine environments, i.e. identical machines (Pm), uniform machines (Qm), and unrelated machines (Rm). Any parallel machine scheduling problem is simply classified as Pm-model with identical machine speeds, formally omitting the fact that some authors actually consider Qm- or Rm-models with different machine processing time concepts. This section covers eight scheduling problems summarized in Table 11.

Table 11: Eight parallel machines scheduling problems

machine environment	constraints B, s_j	constraints r_j	constraints $fmls$	$\alpha \mid \beta \mid \gamma$-notation	section
$Pm \mid \cdot \mid \cdot$	-	-	-	$Pm \mid p-batch, b<n \mid \cdot$	2.2.1
	✓	-	-	$Pm \mid p-batch, B, s_j \mid \cdot$	2.2.2
	-	✓	-	$Pm \mid p-batch, b<n, r_j \mid \cdot$	2.2.3
	-	-	✓	$Pm \mid p-batch, b<n, fmls \mid \cdot$	2.2.4
	✓	✓	-	$Pm \mid p-batch, B, s_j, r_j \mid \cdot$	2.2.5
	✓	-	✓	$Pm \mid p-batch, B, s_j, fmls \mid \cdot$	2.2.6
	-	✓	✓	$Pm \mid p-batch, b<n, r_j, fmls \mid \cdot$	2.2.7
	✓	✓	✓	$Pm \mid p-batch, B, s_j, r_j, fmls \mid \cdot$	2.2.8

2.2.1 Pm | p-batch, b < n | ·

This section begins with the basic bounded batch scheduling problem on parallel BPMs, denoted with $Pm \mid p-batch, b<n \mid \cdot$. The mentioned publications exclusively deal with the longest job processing time model (L). See Table 12 for a tabular overview.

Makespan To our best knowledge, no publications exist for the problem $Pm \mid p-batch, b<n \mid C_{max}$.

On-Time Delivery Mönch and Unbehaun (2007) describe three decomposition heuristics that minimize the sum of the absolute deviations of completion times from the due date of all jobs, given that all jobs are assumed to have the same due date.

Cycle Time Chandru et al. (1993a) present a heuristic that minimizes $\sum C_j$.

Table 12: Publications related to Pm | p-batch, b < n | ·

publication	model	method	objective	constraints
Chandru et al. (1993a)	L	H	CT	-
Mönch and Unbehaun (2007)	L	H*	OTD	-

model:{longest job processing time (L)}; **method:**{heuristic (H), special case(s) (*)}; **objectives:**{cycle time (CT), on-time delivery (OTD)}

2.2.2 Pm | p-batch, B, s_j | ·

This section reviews the bounded batch scheduling problem on parallel BPMs with non-identical job sizes, denoted with $Pm \mid p-batch, B, s_j \mid \cdot$. The mentioned publications exclusively deal with the longest job processing time model (L). See Table 13 for a tabular overview.

Makespan Cheng et al. (2012), Cheng et al. (2013), Xu and Bean (2007), and Chang et al. (2004) provide MIP formulations for C_{max} minimization. Cheng et al. (2012) present a polynomial approximation algorithm and Li et al. (2013) and Damodaran and Chang (2008) propose several heuristics. Cheng et al. (2013) describe a method based on ACO, Chang et al. (2004) propose a SA method, and Xu and Bean (2007) describe a GA. Kashan et al. (2008) compare a hybrid heuristic based on a GA with SA. Beyond metaheuristics, Chen et al. (2011) use a clustering algorithm and Shao et al. (2008a,b) describe a neural net to tackle the problem.

On-Time Delivery To our best knowledge, no publications exist for the problem $Pm \mid p-batch, B, s_j \mid \cdot$ with common objectives related to due dates.

Cycle Time Cheng et al. (2012) provide a MIP formulation and a polynomial approximation algorithm for minimizing $\sum C_j$.

Table 13: Publications related to Pm | p-batch, B, s_j | ·

publication	model	method	objective	constraints
Chang et al. (2004)	L	E, MH	C_{max}	-
Xu and Bean (2007)	L	E, MH	C_{max}	-
Damodaran and Chang (2008)	L	H	C_{max}	-
Kashan et al. (2008)	L	MH	C_{max}	-
Shao et al. (2008a,b)	L	H	C_{max}	-
Chen et al. (2011)	L	H	C_{max}	-
Cheng et al. (2012)	L	E, A	C_{max}, CT	-
Cheng et al. (2013)	L	E, MH	C_{max}	-
Li et al. (2013)	L	H	C_{max}	-

model:{longest job processing time (L)}; **method:**{exact method (E), heuristic (H), metaheuristic (MH), approximation algorithm (A)}; **objectives:**{makespan (C_{max}), cycle time (CT)}

2.2.3 Pm | p-batch, b < n, r_j | ·

This section reviews the bounded batch scheduling problem on parallel BPMs with release dates, denoted with $Pm \mid p-batch, b < n, r_j \mid \cdot$. The mentioned publications deal with the longest job processing time model (L), with the exception of the model in (Koehler and Khuller, 2013) that deals with constant processing times (C). See Table 14 for a tabular overview.

Makespan Li et al. (2005b) and Zhang et al. (2005) present PTASs for minimizing C_{max}. Li et al. (2012a) study the problem under online setting, describing a time window look-ahead model that can foresee all the jobs that will arrive in given time segment.

On-Time Delivery Li et al. (2004) present a PTAS for minimizing L_{max}.

Cycle Time To our best knowledge, no publications exist neither for $Pm \mid p-batch, b < n, r_j \mid \sum C_j$ nor for the weighted case $Pm \mid p-batch, b < n, r_j \mid \sum w_j C_j$.

Multiple Objectives Koehler and Khuller (2013) incorporate deadlines in their model with identical processing times. They provide a polynomial DP algorithm for minimizing the number of batches and C_{max} at the same time as well as a a pseudo-polynomial algorithm for a general batch-count-sensitive objective function.

2.2.4 Pm | p-batch, b < n, fmls | ·

This section reviews the bounded batch scheduling problem with incompatible families on parallel BPMs, denoted with $Pm \mid p-batch, b < n, fmls \mid \cdot$. The mentioned publications deal with the family processing time model (F), with the exception of the model in (Li et al., 2012b) that deals with constant processing times (C). See Table 15 for a tabular overview.

Table 14: Publications related to $Pm \mid p\text{-}batch, b < n, r_j \mid \cdot$

publication	model	method	objective	constraints
Li et al. (2004)	L	A	OTD	-
Li et al. (2005b)	L	A	C_{max}	-
Zhang et al. (2005)	L	A	C_{max}	-
Li et al. (2012a)	L	A#	C_{max}	-
Koehler and Khuller (2013)	C	E*	C_{max}	$\bar{d}_j$

model:{longest job processing time (L), constant processing time (C)}; **method:**{exact method (E), approximation algorithm (A), special case(s) (*), online setting (#)}; **objectives:**{makespan (C_{max}), on-time delivery (OTD)}; **constraints:**{deadlines ($\bar{d}_j$)}

Makespan Uzsoy (1995) presents heuristics for minimizing C_{max}.

On-Time Delivery Uzsoy (1995) presents heuristics for minimizing L_{max}. Balasubramanian et al. (2004) present a GA and Almeder and Mönch (2011), Raghavan and Venkataramana (2006), Mönch and Almeder (2009) propose ACO methods for minimizing $\sum w_j T_j$. Almeder and Mönch (2011) additionally present a VNS scheme and compare both VNS and ACO with a GA. Li et al. (2012b) study the online scheduling of jobs with equal processing times in order to maximize the weighted number of early jobs, given that preemption is allowed.

Cycle Time Uzsoy (1995) presents heuristics for minimizing $\sum w_j C_j$.

Real-Time Control Habenicht and Mönch (2003) investigate the performance of different dispatching and scheduling heuristics in a stochastic environment in order to minimze $\sum w_j T_j$.

Table 15: Publications related to $Pm \mid p\text{-}batch, b < n, fmls \mid \cdot$

publication	model	method	objective	constraints
Uzsoy (1995)	F	H	C_{max}, OTD, CT	-
Habenicht and Mönch (2003)	F	RTC	OTD	-
Balasubramanian et al. (2004)	F	H, MH	OTD	-
Raghavan and Venkataramana (2006)	F	MH	OTD	-
Mönch and Almeder (2009)	F	MH	OTD	-
Almeder and Mönch (2011)	F	MH	OTD	-
Li et al. (2012b)	C	A#*	OTD	*prmpt*

model:{constant processing time (C), family processing time (F)}; **method:**{heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*), online setting (#)}; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD)}; **constraints:**{preemption (*prmpt*)}

2.2.5 $Pm \mid p\text{-}batch, B, s_j, r_j \mid \cdot$

This section reviews the batch scheduling problem on parallel BPMs with non-identical job sizes and release dates, denoted with $Pm \mid p-batch, B, s_j, r_j \mid \cdot$. The mentioned publications deal with the longest job processing time model (L), with exception of the model in (Ozturk et al., 2012) that deals with constant processing times (C). See Table 16 for a tabular overview.

Makespan Chung et al. (2009) and Wang and Chou (2010) define MIP models to find the minimum C_{max} and Li (2012) presents an approximation algorithm. Chung et al. (2009), Damodaran and Vélez-Gallego (2010) and Chen et al. (2010) propose heuristics. Chen et al. (2010) present a GA and a method based on ACO and Wang and Chou (2010) compare a GA with SA. Damodaran and Vélez-Gallego (2012) evaluate a SA approach and a GRASP (Damodaran et al., 2011). Ozturk et al. (2012) consider the problem in the context of hospital sterilization services with equal processing times, presenting a MIP formulation and a approximation scheme as well as polynomial algorithms for two special cases.

On-Time Delivery To our best knowledge, no publications exist for the problem $Pm \mid p-batch, B, s_j, r_j \mid \cdot$ with common objectives related to due dates.

Cycle Time To our best knowledge, no publications exist neither for $Pm \mid p-batch, B, s_j, r_j \mid \sum C_j$ nor for the weighted case $Pm \mid p-batch, B, s_j, r_j \mid \sum w_j C_j$.

Multiple Objectives Xu et al. (2013) propose an ACO method in order to minimize the bi-criteria objective consisting of C_{max} and L_{max}.

Real-Time Control Sahraeian et al. (2014) incorporate size-dependent setup times and compare different heuristics in a stochastic environment in order to minimize C_{max}.

Table 16: Publications related to Pm | p-batch, B, s_j, r_j | ·

publication	model	method	objective	constraints
Chung et al. (2009)	L	E, H	C_{max}	-
Chen et al. (2010)	L	H, MH	C_{max}	-
Damodaran and Vélez-Gallego (2010)	L	H	C_{max}	-
Wang and Chou (2010)	L	E, MH	C_{max}	-
Damodaran et al. (2011)	L	MH	C_{max}	-
Damodaran and Vélez-Gallego (2012)	L	MH	C_{max}	-
Li (2012)	L	A	C_{max}	-
Ozturk et al. (2012)	C	E, E*, A	C_{max}	-
Xu et al. (2013)	L	MH	MO	-
Sahraeian et al. (2014)	L	RTC	C_{max}	-

model:{longest job processing time (L), constant processing time (C)}; **method:**{exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*)}; **objectives:**{makespan (C_{max}), multiple objectives (MO)}

2.2.6 Pm | p-batch, B, s_j, fmls | ·

This section reviews the bounded batch scheduling problem on parallel BPMs with incompatible families and non-identical job sizes, denoted with $Pm \mid p-batch, B, s_j, fmls \mid \cdot$. The mentioned publications exclusively deal with the family processing time model (F). See Table 17 for a tabular overview.

Makespan Koh et al. (2004) present heuristics and a GA for minimizing C_{max}.

On-Time Delivery To our best knowledge, no publications exist for the problem $Pm \mid p-batch, B, s_j, fmls \mid \cdot$ with objectives related to due dates.

Cycle Time Koh et al. (2004) present heuristics and a GA for minimizing $\sum C_j$ or $\sum w_j C_j$.

Multiple Objectives Payman and Leachman (2010) propose algorithms based on Linear Programming (LP) and Integer Programming (IP) and a heuristic-based algorithm in order to simultaneously improve multiple short-term production targets, while considering secondary resources.

Table 17: Publications related to Pm | p-batch, B, s_j, fmls | ·

publication	model	method	objective	constraints
Koh et al. (2004)	F	H, MH	C_{max}, CT	-
Payman and Leachman (2010)	F	H	MO	*sr*

model:{family processing time (F)}; **method:**{heuristic (H), metaheuristic (MH)}; **objectives:**{makespan (C_{max}), cycle time (CT), multiple objectives (MO)}; **constraints:**{secondary resources (*sr*)}

2.2.7 Pm | p-batch, b < n, r_j, fmls | ·

This section reviews the bounded batch scheduling problem on parallel BPMs with incompatible families and release dates, denoted with $Pm \mid p-batch, b < n, r_j, fmls \mid \cdot$. The mentioned publications deal with the family processing time model (F), with the exception of the model in (van der Zee et al., 1997) that deals with constant processing times (C). See Table 18 for a tabular overview.

Makespan To our best knowledge, no publications exist for the problem $Pm \mid p-batch, b < n, r_j, fmls \mid \sum C_{max}$.

On-Time Delivery Malve and Uzsoy (2007) present a GA and Chang et al. (2013) propose a method based on PSO to minimize L_{max}. Kim et al. (2010) present several heuristics for $\sum T_j$ minimization. Bar-Noy et al. (2009) consider the problem with deadlines and present an approximation algorithm maximizing the weight of the scheduled jobs. Mönch et al. (2006b) present a heuristic in order to minimize $\sum w_j T_j$. Mönch et al. (2005) propose a GA and Chiang et al. (2010) describe a memory-based algorithm incorporating concepts from GAs for the same problem. Klemmt et al. (2009) provide a MIP formulation and a VNS scheme for $\sum w_j T_j$ minimization with machine eligibility constraints (M_j). Li et al. (2008, 2009a) present an ACO method that additionally considers sequence-dependent setup times (and M_j).

Cycle Time Tajan et al. (2012) provide a mathematical formulation and a DP scheme for the problem of minimizing the mean CT.

Multiple Objectives Reichelt and Mönch (2006) propose a method based on GAs that aims to optimize $\sum w_j T_j$ and C_{max} at the same time.

Real-Time Control Sha et al. (2004, 2007) present a look-ahead batch dispatching rule taking due date information into consideration. Habenicht and Mönch (2003) describe a simple heuristic that does not take future lot arrivals into account as well as a GA that does consider future arrivals, focusing on $\sum w_j T_j$. Several look-ahead rules that minimize CTs are proposed by van der Zee et al. (1997, 2001), Fowler et al. (2000), van der Zee (2001), Solomon et al. (2002), Cigolini et al. (2002) and Tajan et al. (2012), where Solomon et al. (2002) additionally consider the status of downstream machines in a sense that minimizing sequence-dependent setup times is desired. Murray et al. (2008) propose a batch scheduling heuristic that considers sequence-dependent setup times and take future arrivals into account, evaluated in a stochastic environment. A total cost function is used to combine two conflicting performance measures into one, i.e. total item queuing time and total machine running time.

2.2.8 Pm | p-batch, B, s_j, r_j, fmls | ·

This section reviews the bounded batch scheduling problem with incompatible families on parallel BPMs subject non-identical job sizes and release dates, denoted with $Pm \mid p-batch, B, s_j, r_j, fmls \mid \cdot$. The mentioned publications exclusively deal with the family processing time model (F). See Table 19 for a tabular overview.

Makespan Klemmt et al. (2008) and Klemmt et al. (2011) examine MIP and simulation-based optimization approaches for C_{max} minimization under machine eligibility and deadline constraints, whereby Klemmt et al. (2011) additionally consider machine breakdown periods in their model.

On-Time Delivery Mathirajan and Sivakumar (2006a) present a few greedy heuristics that minimize $\sum w_j T_j$. Klemmt et al. (2011) include machine eligibility constraints, deadlines and machine breakdowns in their models, evaluating MIP and simulation-based optimization for $\sum w_j T_j$ minimization. Gokhale and Mathirajan (2014) allow job splitting in their mathematical formulation for minimizing $\sum w_j T_j$, and present heuristics for it. Kohn and Rose (2012) use VNS to minimize

Table 18: Publications related to $\mathrm{Pm} \mid \text{p-batch}, \mathrm{b} < \mathrm{n}, \mathrm{r_j}, \text{fmls} \mid \cdot$

publication	model	method	objective	constraints
van der Zee et al. (1997)	C	RTC	CT	-
Fowler et al. (2000)	F	RTC	CT	-
van der Zee (2001)	F	RTC	CT	-
van der Zee et al. (2001)	F	RTC	CT	-
Cigolini et al. (2002)	F	RTC	CT	-
Solomon et al. (2002)	F	RTC	CT	*sdst*
Habenicht and Mönch (2003)	F	RTC, MH	OTD	-
Sha et al. (2004, 2007)	F	RTC	OTD	-
Mönch et al. (2005)	F	MH	OTD	-
Mönch et al. (2006b)	F	H	OTD	-
Reichelt and Mönch (2006)	F	MH	MO	-
Malve and Uzsoy (2007)	F	MH	OTD	-
Li et al. (2008)	F	MH	OTD	M_j, *sdst*
Murray et al. (2008)	F	RTC	MO	*sdst*
Bar-Noy et al. (2009)	F	A	OTD	$\bar{d}_j$
Klemmt et al. (2009)	F	E, MH	OTD	M_j
Li et al. (2009a)	F	MH	OTD	M_j, *sdst*
Chiang et al. (2010)	F	MH	OTD	-
Kim et al. (2010)	F	H	OTD	-
Tajan et al. (2012)	F	RTC	CT	-
Chang et al. (2013)	F	MH	OTD	-

model:{constant processing time (C), family processing time (F)}; **method:**{exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A)}; **objectives:**{cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{deadlines ($\bar{d}_j$), machine eligibility (M_j), sequence dependent setup times (*sdst*)}

$\sum T_j$ while considering machine eligibility constraints and deadlines in their optimization model. They analyze the effect of various factors that influence the optimization potential.

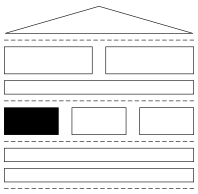
Cycle Time Cakici et al. (2013) present a MIP formulation and a VNS scheme in order to minimize $\sum w_j C_j$. Klemmt et al. (2008) include machine eligibility constraints and deadlines, evaluating MIP and simulation-based optimization for minimizing $\sum C_j$ and $\sum w_j C_j$. Kohn and Rose (2012) consider machine eligibility constraints and deadlines in their optimization model based on VNS. They analyze the effect of various factors that influence the optimization potential when minimizing $\sum C_j$. Kohn and Rose (2013) study the impact of accuracy in lot arrival prediction on the objective $\sum C_j$. Their model based on VNS incorporates machine eligibility constraints and deadlines.

Multiple Objectives Li et al. (2009b) present an ACO method that optimizes $\sum w_j T_j$ and C_{max} at the same time, considering machine eligibility constraints and sequence-dependent setup times. Yugma et al. (2008) include machine eligibility constraints, precedence constraints and deadlines in their model. They propose a method based on LS and SA to simultaneously improve the total number of moves, the batching coefficient and the x-factor. Kohn et al. (2013) experimentally examine the relationship between various objectives and performance measures related to CT, OTD and THP. The underlying VNS model involves machine eligibility constraints and deadlines.

Table 19: Publications related to $\mathrm{Pm} \mid \text{p-batch}, \mathrm{B}, \mathrm{s_j}, \mathrm{r_j}, \text{fmls} \mid \cdot$

publication	model	method	objective	constraints
Mathirajan and Sivakumar (2006a)	F	H	OTD	-
Klemmt et al. (2008)	F	E, H	C_{max}, CT	M_j, $\bar{d}_j$
Yugma et al. (2008)	F	MH	MO	M_j, $\bar{d}_j$, *prec*
Li et al. (2009b)	F	MH	MO	M_j, *sdst*
Klemmt et al. (2011)	F	E, H	C_{max}, OTD, CT	M_j, $\bar{d}_j$, *brkdwn*
Kohn and Rose (2012)	F	MH	OTD, CT	M_j, $\bar{d}_j$
Cakici et al. (2013)	F	E, MH	CT	-
Kohn and Rose (2013)	F	MH	CT	M_j, $\bar{d}_j$
Kohn et al. (2013)	F	MH	MO	M_j, $\bar{d}_j$
Gokhale and Mathirajan (2014)	F	E, H	OTD	*jspl*

model:{family processing time (F)}; **method:**{exact method (E), heuristic (H), metaheuristic (MH), }; **objectives:**{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)}; **constraints:**{deadlines ($\bar{d}_j$), machine eligibility (M_j), sequence dependent setup times (*sdst*), job splitting (*jspl*), precedence (*prec*), breakdowns (*brkdwn*)}



3 Wafer Fabrication

Contents

A semiconductor device is defined as a semiconductor product which has electric elements and wiring, made to fulfill specific functions, according to SEMI International Standards (2009). Such complex electric circuits are also known as IC, chip, microchip or die.

An IC contains minimum one semiconductor die, regardless whether it is on the way of fabrication or completed, whether it is on the way of fabrication or completed, it has been diced, it is mounted on some substrate or it is packaged. If it is packaged, the whole package is assumed as a device and a device may have more than one die. Many dies are usually fabricated on a semiconductor substrate at a time which is often referred to as wafer. The wafer is diced after completing the last processing step and before packaging.

Flash-Memories as a specific type of Electrically Erasable Programmable Read-Only Memories (EEPROMs), Dynamic Random Access Memories (DRAMs), Micro Processing Units (MPUs) and Application Specific Integrated Circuits (ASICs) can be considered as few of the most important semiconductor devices (Cogez et al., 2011).

Technology Trends in development and fabrication of semiconductor devices follow Moore's law. Moore's law is the observation that over the history of computing hardware, the number of transistors on semiconductor devices doubles approximately every two years (Moore, 1998).

Ferrell and Pratt (2000) state that for an IC manufacturer to remain continuously competitive, the cost per unit area of manufacturing semiconductor devices must decrease continuously. The competitive drive for cost reduction results in the scaling of semiconductor devices to ever smaller dimensions as well as in the increase of wafer sizes. Both scaling effects have major influence on semiconductor manufacturer production effectiveness.

According to the ITRS structure widths on substrates, e.g. the gate length, will continuously decrease from approximately 20 nm in 2013 down to approximately 7 nm in 2025, with minor differences among Flash, DRAM, MPU and ASIC devices (Cogez et al., 2011).

Simultaneously to the process of shrinking structure widths, the wafer diameter increases continuously and until today has reached 300 mm in modern wafer fabrication facilities (waferfabs). Intel, Samsung, and TSMC announced in May 2008 that they will work together with suppliers and other semiconductor players to develop the 450 mm technology. These three companies scheduled the first production ramp-ups until 2016 (Cogez et al., 2011). Schaller (2004) presents a case study of the ITRS that investigates technological innovations in the semiconductor industry.

Fabrication The semiconductor device manufacturing process covers numerous process steps —from a bare silicon substrate to a fully functional semiconductor device. The semiconductor manufacturing process is commonly structured into five phases: *a*) wafer fabrication, *b*) wafer test (wafer probe), *c*) assembly, *d*) packaging, and *e*) final test. The wafer fabrication and the wafer test are often summarized with the term frontend. Correspondingly, the term backend frames the assembly, packaging, and the final test.

Wafer fabrication summarizes all processing steps creating fully functional dies out of a bare silicon substrate. The wafer test comprises several testing procedures that ensure faultless dies, respectively identifies defect dies on the wafer. After testing wafers are diced and go through assembly. At the end of which each die is wired and packed. The resulting semiconductor devices are then exposed to high temperatures (aged) and finally tested again, before they are shipped

to the customer. See Figure 4 for a flow diagram visualizing the basic phases in semiconductor manufacturing.

This work exclusively focuses on wafer fabrication, respectively the frontend.

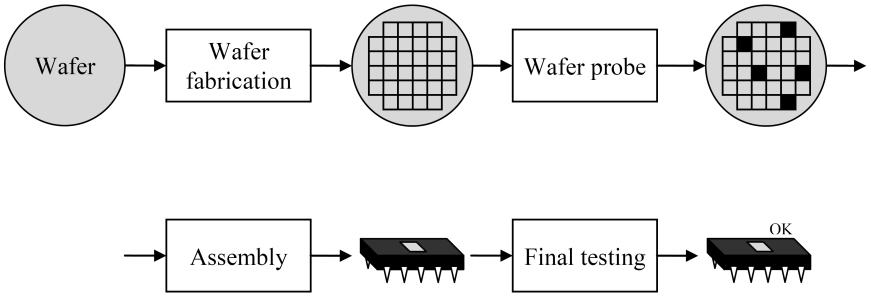


Figure 4: Flow diagram of semiconductor manufacturing (Frantsuzov, 2011); cf. (Mönch et al., 2009)

Facility Hopp and Spearman (2001) define basic terms in manufacturing. A workstation is a collection of one or more machines that perform (essentially) identical functions. The term workstation is also often referred to as work center or CMS, whereas the latter puts focus on identical functions provided by the machines. A routing defines the sequence of process steps provided by workstations passed through by a product or part. In wafer fabrication a part relates to a carrier, also known as cassette or lot. A carrier or cassette describes an open structure that holds one or more substrates (SEMI International Standards, 2009). SEMI E11 (2000) provides carrier specifications for 125 mm, 150 mm, and 200 mm plastic and metal carrier. The term Front Opening Unified Pod (FOUP) refers to a closed structure that holds 300 mm wafers; (SEMI E47, 2000) provides specifications for FOUPs. An order is a request from a customer for a particular product, in a particular quantity, to be delivered on a particular date. A job in wafer fabrication refers to a single lot at a certain stage of production, waiting for the next process step to be carried out on a suitable equipment as a part of a work center. A scheduler or dispatcher system controls the material flow in a waferfab, assigning jobs to machines at any point in time.

Contamination Control Semiconductor devices are very vulnerable to many types of contaminants. Especially particles coming from workers, generated by equipment and present in processing chemicals, create risk of defects on the device. Defects occur when particles located at critical areas on the wafer surface destroy the device functioning, for example by interrupting electrical signals or altering electrical properties.

With respect to high production yields semiconductor manufacturer pursue a total cleanroom strategy. A proper cleanroom design and a filtering technology provide particle-poor air with constant environmental conditions for production. These controlled environmental conditions include the temperature, pressure, humidity of the air and the composition of the gases (van Zant, 2004). Compared to the outdoor air, the shop floor is under constant overpressure, preventing any particle or gas from uncontrolled penetrating the cleanroom. Workers get entry and exit through air locks and wear special clothing covering the body. The shop floor is constantly fed with clean air through complex filtering systems at the ceiling. At the same time the cleanroom air is constantly extracted by continuous suction through a perforated floor. In consequence a laminar air flow from the ceiling to the floor is established, preventing particles to hover through the cleanroom. The number of particles allowed in cleanrooms is defined by the International Organization for Standardization (ISO) in ISO 14644-1 (1999). Today wafer fabrication facilities using non-closed wafer carriers require the cleanroom standard ISO 1. ISO 1 ensures that one cubic meter of air does not count not more than 10 particles equal or larger than 0.1um and not more than two particles equal or larger than 0.2um (Ferrell and Pratt, 2000).

Computer Integrated Manufacturing (CIM) Modern semiconductor manufacturing systems manage interoperability of tens of dozens of sub-systems that fulfill specific functions as a part of the Computer Integrated Manufacturing (CIM) framework. From technical view, the CIM framework in terms of a technical system is often referred to as MES.

A typical factory network combines *a*) the AMHS, *b*) production equipment, *c*) scheduler and/or dispatcher system, *d*) and many other factory systems (Ferrell and Pratt, 2000). Figure 5 depicts a conceptual view of the factory system.

In order to ensure interoperability among these systems, standards have been developed that provide guidelines for system architecture and communication. SEMI E81 (2000) describes a CIM framework as a software infrastructure that creates a common environment for integrating applications and sharing information in a semiconductor factory. SEMI E96 (2000) defines standards for the technical architecture that enable application components to cooperate in a CIM/MES environment, needed for an improved component interoperability, substitutability and extensibility.

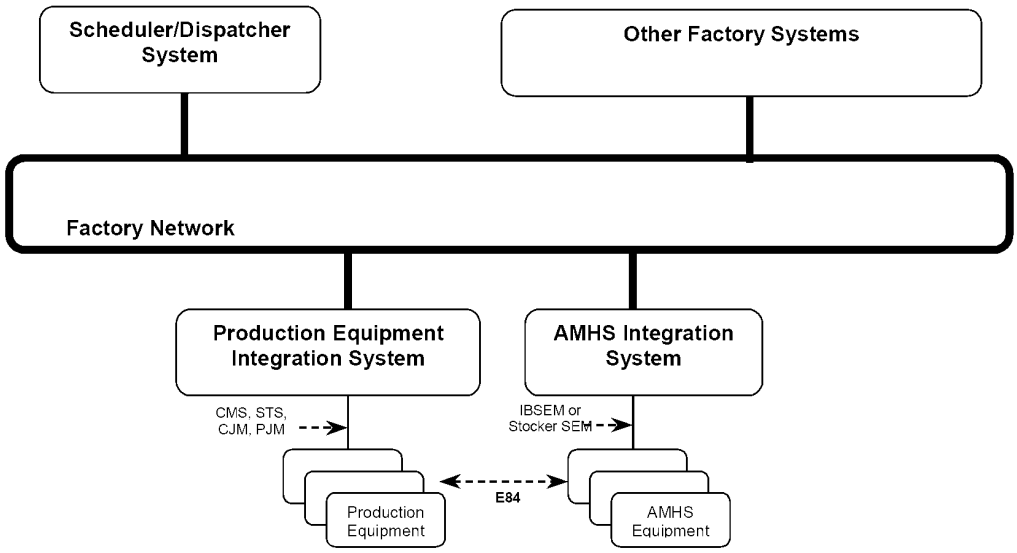


Figure 5: Conceptual view of the factory system (Ferrell and Pratt, 2000)

3.1 Unit Processes

Semiconductor manufacturing consists of a series of sequential process steps. The process flow in Complementary Metal Oxide Semiconductor (CMOS) technology counts 600 to 1200 single process steps, still rising. Starting with raw wafers as the basic raw material, the chip grows with each layer, also called photo-layer. After processing each wafer contains hundreds of identical rectangular chips. The chips are separated by sawing or laser cutting. The various processing steps fall into five general categories: *a*) film formation, *b*) lithography, *c*) etching, *d*) impurity doping, and *e*) non-value processes.

Figure 6 illustrates the interrelationship between these major process categories. This cyclic process flow is the reason for reentrant flows in a waferfab (May and Spanos, 2006). Semiconductor Manufacturing Technology (SEMATECH), an association of member companies cooperating in key areas of semiconductor technology, presents a 300 mm aluminum process flow for 180 nm technology, covering six metal layers, 21 masks, 43 tool types, and 316 steps. The raw process time for this process flow is 8.9 days (Campbell and Ammenheuser, 2000).

3.1.1 Film Formation

Film formation describes any process that modifies the wafers' surface. Many different kinds of thin films are used to fabricate semiconductor devices, including thermally grown oxide films, deposited dielectric films and deposited metal films. Deposition is any process that grows, coats

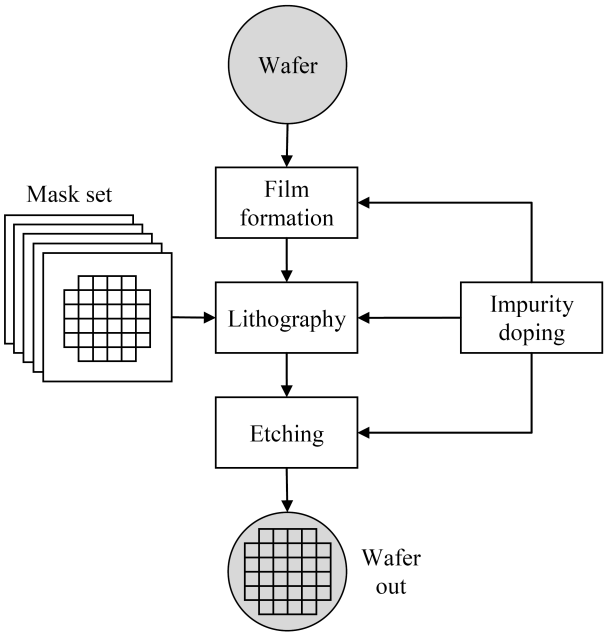


Figure 6: Flow diagram for a generic IC process sequence (May and Spanos, 2006)

or otherwise transfers material onto the wafer. Thermal oxidation processes create an oxide layer on the wafer surface. Planarization that belongs to the removal processes also modifies the wafer surface by partly removing films and thus falls into this category. Film formation is often followed by photolithography or impurity doping (May and Spanos, 2006).

Thermal Oxidation A dioxide film functions as an insulator in a number of device structures or as a barrier to diffusion or implantation during device fabrication. Semiconductors can be oxidized by various methods, and among these, thermal oxidation is the most important for silicon devices. Thermal oxidation employs oxidants under high temperature to transform (oxidize) a bare silicon surface to silicon dioxide, which is actually grown out of the substrate surface. Commonly there exist three types of equipment for this kind of process: *a*) the vertical furnace, *b*) the horizontal furnace, and *c*) the Rapid Thermal Processor (RTP); whereas vertical furnaces play the most important role (May and Spanos, 2006). Singh et al. (2003) point the reduction of CT and process activation energy as the two distinct advantages of RTP over CFP, respectively vertical and horizontal batch furnaces.

Deposition Deposition is a process that transfers a metal material onto the wafer in thin films. Most common depositing technologies rely on the principles of physical, chemical or electrochemical vapor deposition. Physical Vapor Deposition (PVD) of metals is mostly accomplished by *sputtering* today. Sputtering is a process in which ions of an inert gas such as argon are electrically accelerated in a high vacuum toward a target of pure metal, such as tantalum or copper. Upon impact, the argon ions sputter off the target material, which is then deposited as a thin film on the silicon wafer. Chemical Vapor Deposition (CVD) is the traditional method used to deposit dielectric films on wafers. CVD occurs when a gas mixture is passed over a heated substrate and chemical reactions are initiated. The process is conducted in CVD reactors and can be performed at atmospheric pressure or at low pressure. Plasma-Enhanced Chemical Vapor Deposition (PECVD) uses high energetic plasma to initiate the chemical reaction. Electrochemical Deposition (ECD) is a wet chemistry process that is often used to build conductive wires on the wafer (May and Spanos, 2006).

Planarization Chemical-Mechanical Polishing (CMP) is used to create a planar surface on the wafer. This may be necessary in order to set up the wafer for the next processing steps. The process

uses a chemical slurry in conjunction with a polishing pad. The pad is pressed on the wafer and rotated with different axes. This removes material and tends to even out any irregular topography, making the wafer flat or planar (May and Spanos, 2006).

3.1.2 Photolithography

The processes of pattern transfer and pattern generation are generally referred to as photolithography. In lithography a structure from a mask is transferred on the substrate by the use of a photoresist. Simplified, the lithography process contains three steps: *a*) coating, *b*) exposure, and *c*) development. Between these three main steps the wafer goes through several bake-steps heating the wafer in order to harden the photoresist. At the beginning, the wafer is coated with a chemical liquid called photoresist, which is spun onto the wafer surface. Then the photoresist is selectively exposed to ultraviolet light in an optical lithographic system, transferring the structure of a mask onto the wafer surface. The photoresist development is usually done by flooding the wafer with the developer solution. If a positive photoresist is used, the exposed regions of the resist are dissolved in the developer, whereas the unexposed regions remain. As a result, an exact copy of the mask structure is formed by the remaining photoresist on the surface. Photolithography is generally followed by etching, which in turn is often followed by another impurity doping or film formation process. Finally, the remaining photoresist is removed from the surface, stripped away by a chemical liquid or burned to ashes by an oxygen plasma (May and Spanos, 2006).

3.1.3 Etching

After the photolithography process, a specific pattern made of photoresist covers the wafer surface. To produce complex electric circuits, these resist patterns must be transferred into the underlying layers. The pattern transfer is accomplished by an etching process that selectively removes unmasked portions of a layer. Etching processes are conducted in a wet or dry ambient. Wet etching processes use liquid etchants to remove material from the surface, whereas the wafer undergoes a sequence of baths with liquid etchants. Dry etching uses plasma to remove material from the wafer surface. Wet chemical etching is used extensively in semiconductor processing, but modern wafer fabrication process sequences avoid wet etching and use plasma etching instead. The etching process is often followed by impurity doping (May and Spanos, 2006).

3.1.4 Impurity Doping

Doping technologies are used to modify electrical properties of the substrate. Diffusion and ion implantation are the two key methods of impurity doping. Originally diffusion processes were used to bring dopants into the substrate. Later ion implantation is conducted to modify electrical properties (May and Spanos, 2006).

Diffusion The diffusion process is conducted in a furnace. Under high temperature atoms from another material diffuse into the wafer surface. The diffusion effect is accomplished by placing semiconductor wafers in a high-temperature furnace and passing a gas mixture that contains the desired dopant through it. This step is followed by a drive-in diffusion process —a thermal annealing process which serves to activate the implanted dopants (May and Spanos, 2006).

Ion Implantation In the ion implantation process the energetic dopant ions are implanted into the semiconductor by means of an ion beam. Ions of a certain material are accelerated to a high energy level in an electrical field and impacted into the wafer. The main advantages of ion implantation are its more precise control, its reproducibility of impurity dopings and its lower processing temperature compared with those of the diffusion process (May and Spanos, 2006).

3.1.5 Non-Value Processes

Non-value adding process steps are the key factors to ensure quality and trace yields. Since particles on a wafer surface may lead to defect chips, the process flow involves cleaning processes decreasing

the number of particles on the wafer surface. Process steps in metrology ensure that processes follow their specifications (May and Spanos, 2006).

Cleaning Cleaning processes are needed to keep the surface clean from particles. The wafer undergoes a sequence of baths with liquids in order to remove particles from the surface. After cleaning particle-measuring processes are conducted. Cleaning is performed especially before proceeding high-temperature processes (May and Spanos, 2006).

Metrology Metrology processes are the key factors to monitor the quality of value-adding processes. Process monitoring enables operators and engineers to detect problems early on to minimize their impact and thus ensures producing reliable, high-quality devices repeatably. Manufacturing line monitors consist of extremely sophisticated metrology equipment that can be divided into tools characterizing the state of features on the semiconductor wafers themselves and those ones that describe the status of the fabrication equipment operating on those wafers. Equipment state measurements ensure that the process equipment works as desired. The measurements, performed on wafers after a process, characterize physical parameters, such as film thickness, uniformity and feature dimensions; or electrical parameters, such as resistance and capacitance. Based on these observations, it is possible to derive appropriate actions that help to adjust the process equipment. Wafer state measurements are conducted multiple times in a process flow in order to detect problems and ensure high quality. Such investigations include visual inspections as well as sophisticated physical and electrical measurements of various characteristics that describe the state of a wafer (May and Spanos, 2006).

3.2 Process Equipment

Generally an equipment is a mechanical entity in the factory which plays a role in the manufacturing process, i.e. for processing, transport, and/or storage of material. As long as the focus lies on on wafer fabrication, the term equipment is used as a synonym for machines processing wafers, also called wafer fabrication equipment or process equipment (SEMI International Standards, 2009).

During the years, numerous equipment architectures have been developed and introduced into waferfabs. Semiconductor manufacturer make use of tens of dozens of different equipment types in a single facility. Each equipment is designed to fulfill a particular function, thought for modifying wafer surface or layers by physical or (electro)chemical reactions. The architecture of a specific equipment type is often dictated by the process, which is intended to be carried out by the equipment. Table 20 shows a mapping between unit processes and equipment types. Mönch et al. (2011a) presents a similar mapping between common process types and certain processing characteristics, such as the equipment type or area specific constraints.

From the scheduling expert's view, the basic principle the equipment processes lots/wafers is of particular interest, especially internal wafer flows. The job processing behavior may substantially differ among equipment types, since they represent different construction schemes, which result in varying THP rates and processing times.

Table 20: Mapping between unit processes and equipment types

process type		equipment type			
process group	unit process	single-wafer	batch furnace	wet bench	cluster tool
film formation	thermal oxidation	x	x (CFP)		
	deposition				x
	planarization				x
lithography					x
etching	dry				x
	wet	x		x	
impurity doping	diffusion	x (RTP)	x (CFP)		
	ion implantation	x (MBP)			
non-value	cleaning	x		x	
	metrology	x			

SEMI Standards Despite of these differences, but also exactly for that reason, Semiconductor Equipment and Materials International (SEMI) provides a set of standards. SEMI E10 (2000) contains specifications for measuring reliability, availability and maintainability performance in order to establish a common basis for communication between users and suppliers of semiconductor manufacturing equipment. SEMI E30 (2000) provides a Generic Model for Communications and Control of Manufacturing Equipment (GEM). SEMI E4 (2000) contains the SEMI Equipment Communications Standard Part 1 (SECS-I) that defines a communication interface suitable for the exchange of messages between semiconductor processing equipment and a host. This standard does not define the data contained within a message. For that purpose, the SEMI Equipment Communications Standard Part 2 (SECS-II) defines the details of the interpretation of messages in (SEMI E5, 2000).

3.2.1 Single-Wafer Processing Equipment

The SWP equipment commonly stands for types of wafer fabricating equipment that process wafers individually. The single wafer equipment is capable of processing one single wafer at a time. Consequently wafers are processed in a strict sequential manner —one after another. The sequential processing scheme holds for processing jobs the same. The single wafer equipment provides short CTs due to its architecture. This is especially beneficial for waferfabs dealing with small lot sizes. Depending on the type of process, there exists additional wafer capacity inside the equipment, for example for cooling or heating functions. Cleaning processes, measurements and rapid thermal processes are usually performed in this type of equipment.

This equipment usually provides two load ports (19), one for the job in process and one for buffering the next job for process. The Equipment Front End Module (EFEM) (15) includes an atmospheric robot (20) for moving wafers between load ports and single wafer load locks (16A, 16B). The EFEM can also be fitted with a wafer aligner (21), used to detect the wafer orientation (notch). The reactor (13) can perform a process for a single wafer at a time. Figure 7 illustrates the basic architecture of an SWP equipment. Singh et al. (2003) concern with the impact of SWP on semiconductor manufacturing. They particularly point out the reduction of CT as distinct advantage of SWP compared to BP. In the same line Stubbe (2010) regards the conversion of batch processes to mini-batch or single-wafer processes in combination with small lot sizes as a promising strategy to reduce CTs. Moslehi et al. (1992) present an overview of various SWP techniques.

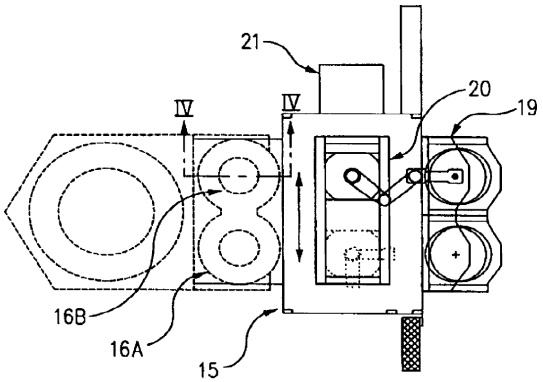


Figure 7: Single wafer processing equipment architecture (Stevens and Jakubiec, 2002)

3.2.2 Cluster Tool

The term cluster tool is widely spread and describes an equipment type that combines a number of process modules together, combining several process modules to a single system. Figure 8 illustrates the basic architecture. A cluster tool effectively combines a number of process chambers to a single machine. The typical (circular) cluster tool is basically composed of a mainframe and an attached EFEM. The mainframe consists of a central wafer handling robot connected to the process chambers and a number of load locks offering access to the mainframe. The EFEM comprises a

number of load ports and a wafer handler, which enables the wafer transfer between load ports and load locks.

The procedure performed to process a lot follows a defined sequence of activities. After the carrier is put on the load port, the EFEM's handler sequentially transfers the wafers to the load lock, which then pumps to a vacuum. Thereafter the mainframe's handling robot transports one wafer after another to a free process chamber. According to the recipe, a wafer possibly visits more than one chamber before it eventually returns to the load lock. At the time when the last wafer of a lot returns to the load lock, the load lock vents to the atmosphere. Lastly the EFEM's handler transfers processed wafers back to the carrier. Finally the carrier is removed from the load port in order to continue with the next operation.

Obviously cluster tools are made to process lots in parallel, since they comprise multiple load locks and several chambers, which may offer different processes. But strictly speaking, the internal processing mode, either sequential or parallel, is firstly determined by the combination of lots recipes performed and secondly depending on the internal wafer scheduling policy. The recipe defines the internal wafer routing, meaning the sequence of specific chambers to visit. Different recipes commonly represent different process setups, either manifesting in different wafer routing sequences or at least in the same wafer routing sequences with differing chamber process times.

Cluster tools are typically used to perform CVD, PVD, or CMP processes. Newest architectures count more than one mainframe, combined to a line, where each mainframe is connected to process chambers, also known as linear cluster tools (Park and Morrison, 2011). This kind of equipment internally provides a kind of flow line. In photolithography such complex linear cluster tools combine coating, exposure, developing and baking processes into one fabricating entity. Yi et al. (2007) and van der Meulen (2007) discuss the advantages of linear cluster tools with respect to their effect on THP and CT.

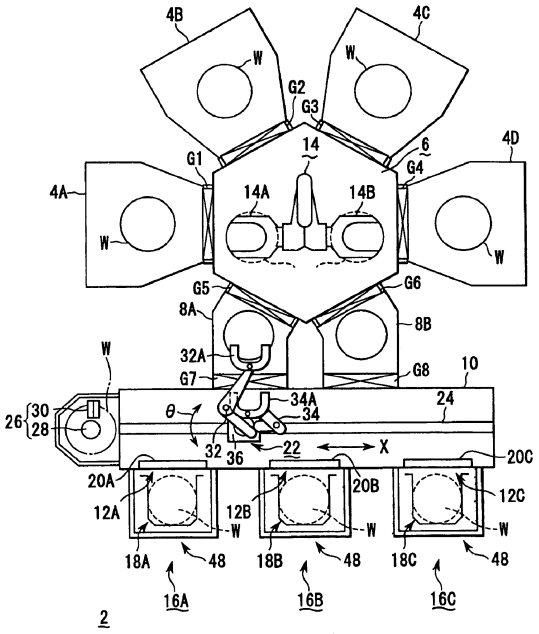


Figure 8: Cluster tool architecture (Yoshida et al., 2007)

3.2.3 Batch Furnace

Batch furnaces may be the most known representatives of equipment performing batch operations. Diffusion and oxidation processes are commonly performed by batch furnaces. A number of lots is formed to a batch, grouped to a single job where all wafers and lots are started together and processed simultaneously by the use of the same process program (recipe). The equipment usually provides a single process chamber performing the heat treatment. An internal wafer handling

system moves wafers from lots on the load ports into the quartz boat, which is then moved into the process chamber. Common quartz boats, respectively batch furnaces, offer space for up to 200 wafers. Batch furnaces commonly provide internal buffer for non-productive wafer needed for processing. There exist horizontal and vertical furnaces and the vertical furnace is the most frequent one. Some variants of semiconductor BP furnaces provide two boats offering the capability to process two batches in parallel, as shown in Figure 9.

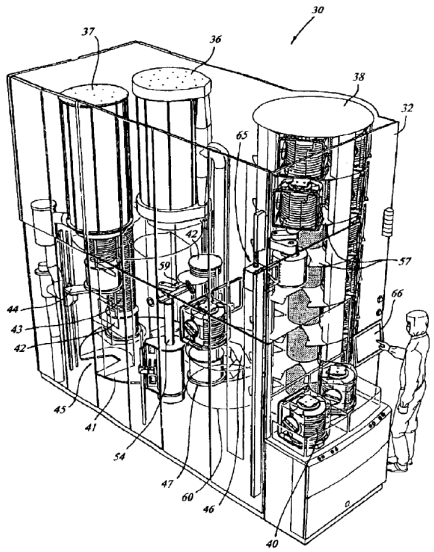


Figure 9: Vertical batch furnace architecture (van den Berg and Den Hartog, 2004)

3.2.4 Wet Bench

A typical wet bench is capable to process a certain number of lots in parallel, whereas the wafers undergo a specific sequence of baths with liquids. A wet bench usually performs etching and cleaning processes. Wet benches provide BP, similar to batch furnaces. Up to four lots are formed to a batch and go through the sequence of tanks together, whereas only one batch occupies a tank at a time. After a lot is loaded on the load port (36), a wafer handling system (34) moves the wafers to an internal wafer carrier (38) that forms the batch. This way, wafers from up to four lots are grouped together in order to proceed the following tank sequence together. An internal carrier handling system (42) transfers the wafer carrier from tank to tank (40). See Figure 10 for an illustrated example.

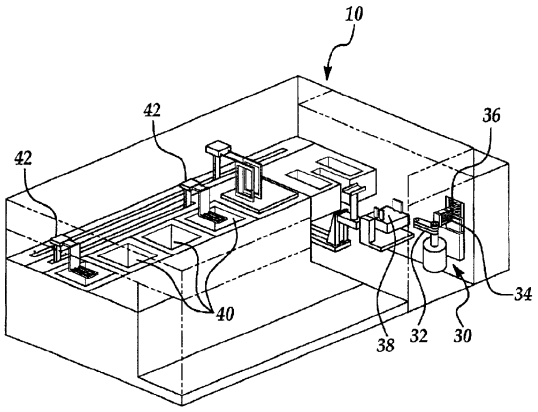


Figure 10: Wet bench architecture (Su et al., 2004)

3.3 Automated Material Handling

The AMHS frames devices for material movement and storage, combining various components to a fully automated system providing transport between process equipment. In the 200 mm wafer era automated wafer handling has limited use in the semiconductor industry. With the shift to 300 mm wafers automation becomes necessary and a key factor to maximize the productivity due to the increased weight and size of 300 mm wafers (Nazzal and Bodner, 2003). The key focus lies on increasing the throughput of transports, reducing the average delivery times and improving the reliability (Cogez et al., 2007). In (Cogez et al., 2011) ITRS additionally recommends actions that focus on a more interactive control for an accurate scheduled delivery. Agrawal and Heragu (2006) also point out that the newer 300 mm waferfabs place a high level of emphasis on AMHSs as important tools to reduce CTs; they discuss various approaches for automated material handling in waferfabs.

Automating the wafer transport system in waferfabs involves several levels of automation. Nazzal and Bodner (2003) distinguish between four types of automation: *a*) material storage, *b*) automated lot transport, *c*) equipment automation, and *d*) Material Control System (MCS) coordinating the efforts of the various automation systems.

SEMI (2000) provides an overview of factory automation requirements and design. The examples in this document have been applied in a working semiconductor 200 mm waferfab. Accordingly, Heinrich and Deutschländer (2012) discuss challenges, present summarized rules and show the advantages of automation for a low-volume high-mix 200 mm facility.

3.3.1 Storage, Transport and Equipment Automation

Wafer fabrication factories will require the baseline capabilities of stocker storage and material transport. These stocker and transport systems will be required to be fully integrated with each other and the factory MES in order to realize the full vision of cost effective automated material transport to and from production equipment (SEMI E102, 2000). AMHS storage equipment, also known as *stocker*, is a mini environment offering material storage capacity for regular use, and in case of cleanroom exceptions as safety device. Nazzal and Bodner (2003) list common technologies for transporting lots: *a*) Overhead Hoist Transport (OHT), *b*) Continuous Flow Transport (CFT), *c*) Automated Guided Vehicles (AGVs), *d*) Rail Guided Vehicles (RGVs), and *e*) Personnel Guided Vehicles (PGVs).

OHT is established in facilities where Overhead Hoist Vehicles (OHVs) are suspended from ceiling-mounted rail mechanisms and are capable of delivering to/retrieving from stocker ports and process tools from directly overhead. Overhead Shuttles (OHSs) connect stocker equipment, while carrying usually two lots. CFT is established through the use of a conveyor system. AGVs stand for movement platforms with automatic guidance capability and on-board robots for loading/unloading. RGVs name automated vehicles that move in a straight line along a fixed path on an in-floor rail. PGVs designate ground based manually moved transporters (see Figure 11).

Typically, AMHSs used in waferfabs are based on discrete vehicle-based overhead systems such as OHVs. Conveyor-based CFT implementations are starting to gain support with the expectations that CFT systems will be capable of handling high-volume manufacturing transport requirements (Nazzal and El-Nashar, 2007). Heinrich et al. (2008) discuss the advantages of a CFT system with respect to automation and its extendability to direct tool loading. Beyond wafer fabrication, Vis (2006) discusses literature related to design and control issues of AGV systems at manufacturing, distribution, transshipment and transportation systems.

SEMI Standards Semiconductor factories equipped with an AMHS require an integrated software system to realize automated material movement. This AMHS integration system must be interoperable with AMHS equipment, production equipment integration systems and other factory systems. In order to achieve this goal, the AMHS integration system must conform to standard communication protocols, state models and interfaces (Ferrell and Pratt, 2000). Refer to Figure 5 for a conceptual view of a factory system. SEMI E88 (2000) establishes a Specific Equipment Model (SEM) for AMHS storage equipment. SEMI E84 (2000) and SEMI E23 (2000) provides guidelines and defines communications associated with the material hand-off operations between the production equipment and the components of the AMHS, for example AGV, RGV and OHV.

SEMI E106 (2000) focuses on the complex interdependencies among the the SEMI standards for 300 mm physical interfaces and carriers. SEMI E101 (2000) provides a functional structure model of an EFEM that handles carriers and substrates at the interface between the AMHS and the process equipment.

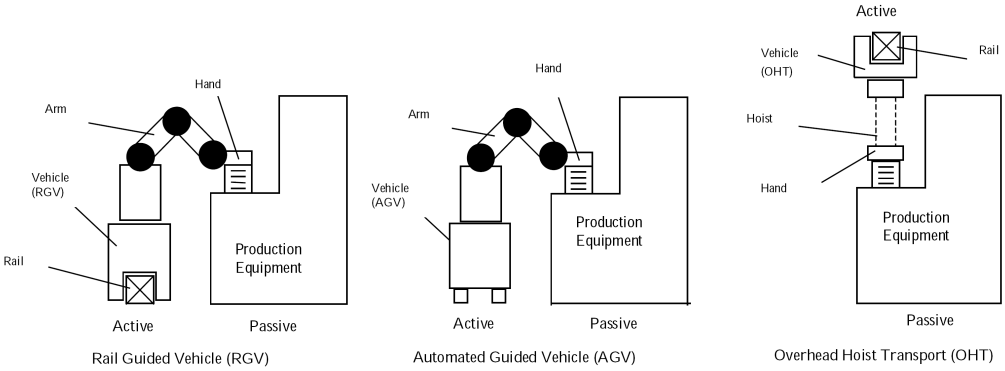


Figure 11: Examples of AMHS equipment (SEMI E84, 2000)

3.4 Factory Layout

In a typical wafer fab, there often are dozens of process flows and several hundred machines. Machines are expensive, ranging in price from a couple of hundred thousand dollars to over thirty million dollars per tool. The economic necessity to reduce capital spending dictates that such machines are shared by all jobs requiring the particular processing operation provided by the machine. This results in a manufacturing environment that is characterized by reentrant flows (Mönch et al., 2011a).

Layout and modeling results show that the size of 300 mm factories may be significantly larger than of current 200 mm factories. This footprint is highly dependent on the layout chosen (Quinn and Bass, 1999). The layout study is important because the layout largely determines the initial investment and production efficiency of a plant compared with other downstream activities. While approximately one billion U.S. dollars was necessary for a new semiconductor fab in 1995, manufacturers today need to invest two to three times more for the same type of facility. In addition, layout is difficult and expensive to modify once it is set up (Chung and Jang, 2007).

There exists a variety of layout strategies in production, giving a scheme how to organize equipment on the shop floor. These layouts either orientate on the process or on the product or provide a trade-off between both. Four basic layout strategies emerged within the last 20 years: *a*) farm layout, *b*) serial layout, *c*) cellular layout, and *d*) ballroom layout.

Drira et al. (2007) provide a recent survey on layout problems found in several types of manufacturing systems. They suggest a general framework to analyze various layout concepts depending on manufacturing system features. Jerbi and Chtourou (2012) particularly compares cthe cellular and functional layout.

3.4.1 Farm Layout

The design of waferfabs for high volume production has traditionally been dominated by the functional or process layout, where work centers consist of groups of similar or identical machines that are capable of performing the same type of unit process. This kind of process-oriented layout is also known as farm layout, bay layout or reentrant layout. The farm layout is still most common due to several advantages. The four most important ones are: *a*) flexibility in scheduling and robustness to machine breakdowns, *b*) lower machine requirements and consequently lower capital investment cost, *c*) less need for clean room space due to close proximity of the machines, and *d*) easier organization of the supply of gases and chemicals on the shop floor in cause of a homogeneous equipment set in a bay. Unfortunately, these advantages also lead to complex,

reentrant product flows (Hase et al., 1994). Appendix D visualizes the material flow around a single work center in the furnace area, showing the set of connected work center that sends to and receives material from the focused work center.

SEMATECH Layout Study In (Quinn and Bass, 1999) and (Campbell and Ammenheuser, 2000) SEMATECH analyzes variants of the farm layout and presents a comparison based on simulation. Three layout configurations have been designed for this project: *a*) Farm, *b*) Hybrid, *c*) and Modified Hybrid.

In the Farm layout all similar tools are placed together in the same bay or set of bays. Thus, there is a separate set of bays to hold the metrology tools. The Hybrid layout is derived from the Farm layout by distributing metrology tools among the bays. If up to three metrology steps follow a step that utilizes a tool within a given bay, the necessary metrology tools are located in that bay. In the Modified Hybrid layout, ashers and wet benches are also distributed among the various bays in addition to the metrology tool redistributions.

Based on their experiments, they propose the Hybrid layout and point out a list of advantages in their simulation studies: *a*) The Hybrid layout outperforms the Farm layout in terms of CT and average WIP level, for average and standard deviation measures. *b*) The Hybrid layout shows the lowest values for standard deviation measured for CT and WIP. *c*) The Modified Hybrid layout shows the best results for average WIP and average CT, outperforming the Hybrid layout in terms of average values, although resulting in the highest values for CT and WIP standard deviation. *d*) Additionally, the Hybrid layout resulted in the smallest footprint, fewest bays and lowest factory area per wafer starts per week. Since simulation runs were made with only one process flow, a single aluminum 180 nm logic process flow with 316 steps, the results could not be directly transferred to a multi-product waferfab.

3.4.2 Serial Layout

The serial layout concept is clearly product-oriented, offering dedicated process equipment for each operation required. This flow line layout yields a simple, linear product flow with short CTs, but leads to extremely high machine requirements (and more clean room space) compared to the reentrant layout. In the light of high capital investments, this layout is unlikely to be a practical alternative (Hase et al., 1994). Nevertheless, there exist recent attempts to establish flow line concepts in wafer fabrication, at least partially in suitable sections, e.g. in the wafer test area (Keil et al., 2011; Eberts et al., 2012).

3.4.3 Cellular Layout

The cellular layout concept groups machines that are dedicated to performing operations related to a fixed number of mask layers. Cellular layouts can be seen as an intermediate stage between the bay layout and the flow line. The idea is to establish partial flow lines for certain layers, whereas an entire mask layer is fabricated in a cell. Consequently one cell offers unit processes for a fixed number of mask layers. They examine the performance of several different cellular and functional layouts using simulation experiments. Despite of the fact that the simulation model is quite simple, e.g. includes only one product, the authors point out the benefits of cellular manufacturing. In particular, they emphasize that cellular layouts requiring only modestly higher capital investment can yield significantly lower CTs in heavily loaded waferfabs. They also expect that the cellular concept results in *a*) reduced setup time due to fewer operations being processed on a given piece of equipment, *b*) yield improvement due to fewer changeovers, *c*) less down time as operators take on maintenance functions, and *d*) simpler material handling due to machines in the cells being in close proximity. But, as a result of their simulation results, they state that the presence of unreliable machinery causes the performance of cellular layouts to deteriorate (Hase et al., 1994, 1997). Similarly, Chang and Chang (1998) propose a layer-based approach that groups the equipment of continuous process layers in the same area or cell. They discuss the layer-based layout approach based on a simulation model that contains one logic product with 16 layers, created in 245 steps.

3.4.4 Ballroom Layout

The ballroom layout is proposed by industry to increase the flexibility and productivity of the traditional bay layout. The ballroom layout involves larger rooms than the bay layout and connects the machines in a room by one combined OHT/OHS loop. This is a functional layout that provides even higher routing and product mix change flexibility than bay-based layouts. This layout increases the direct inter-bay transportation, and reduces the transportation time and the initial investment costs (Chung and Jang, 2007). Refer to Figure 12 visualizing a variant of a ballroom layout.

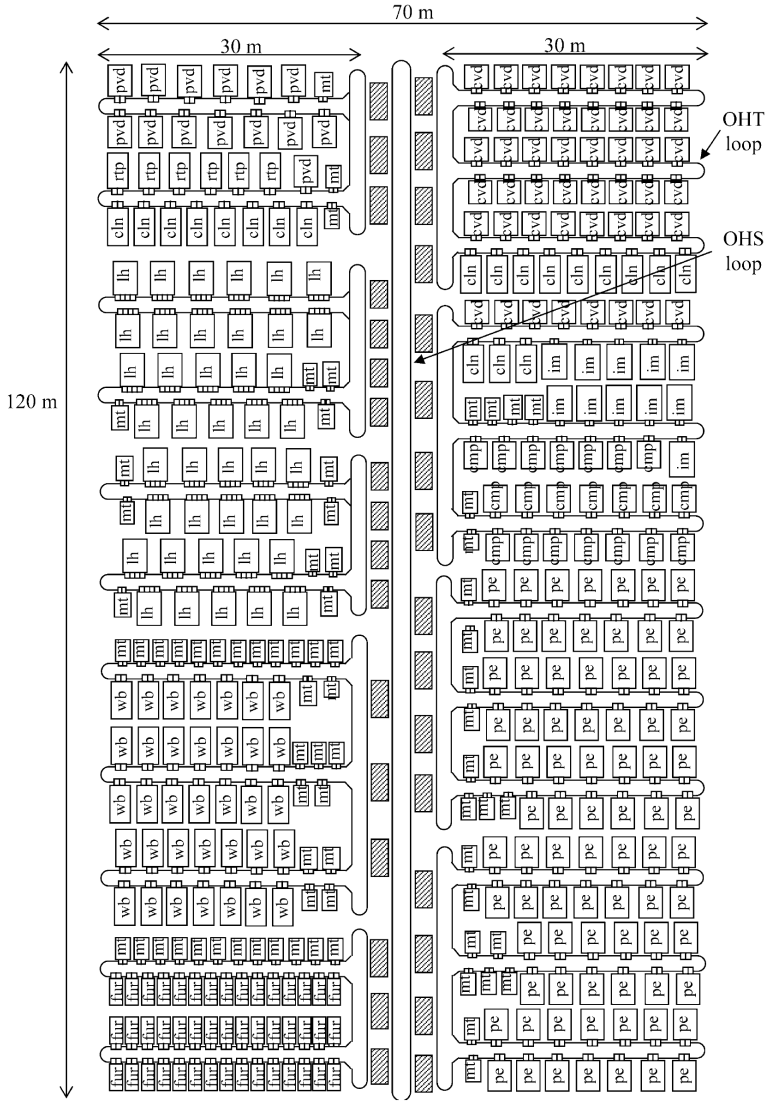
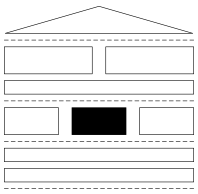


Figure 12: Integrated ballroom layout (Chung and Jang, 2007)



4 Modeling and Simulation

Contents

Modeling and Simulation gains rising importance for almost any kind of industry and science. Informally, Modeling and Simulation (M&S) as a multidisciplinary field in science, provides methods based on computer technology to imitate artificial and/or real systems. The books by Law and Kelton (2000) and Woolfson and Pert (1999) provide early introductions to simulation modeling and analysis. For more recent books that deal with M&S, see (Sokolowski and Banks, 2009; El Sheikh et al., 2008; Chung, 2004). Crosbie (2010) presents a survey on grand challenges in M&S today. Wainer (2009) and Banks (2009) point out various advantages of M&S.

Informally, M&S techniques are used to computationally imitate the behavior of any kind of system. Maier and Rechtin (2000) define a system as a set of different elements so connected or related as to perform a unique function not performable by the elements alone. A system has components, relationships and implicitly a boundary, that separates it from the rest of the environment (Krygiel, 1999). According to ISO/IEC 15288 (2008) a system may be configured with one or more of the following system elements: *a*) hardware, *b*) software, *c*) data, *d*) humans, *e*) processes, *f*) procedures, *g*) facilities, *h*) materials, and *i*) naturally occurring entities.

Law and Kelton (2000) identify a number of ways to study a system: *a*) experimentation with the actual system or with a model of the system, *b*) investigation based on physical or mathematical models, and *c*) modeling approaches based on analytical or simulation solutions (see Figure 13). Among mathematical models, Wainer (2009) further distinguishes between analytical, numerical and simulation approaches.

Advantages and Disadvantages of Simulation In addition to numerous economical benefits, such as reduction in CT of R&D activities, M&S develops an understanding by observing how a system operates (Banks, 2009). Among all advantages, the capability to compress and expand time to allow the user to speed up or slow down the system's behavior can be seen as the most powerful one. Banks (2009) notes to bear in mind that simulation modeling and analysis can be time consuming and requires special training needed for building simulation models. Similarly Shanthikumar et al. (2007) mention that simulation requires an enormous amount of input data and substantial resources to maintain and update the model. Due to the stochastic nature of (most) simulation models, multiple replications are needed to perform a confident statistical analysis. Therefore, it can be difficult and extremely time-consuming to explore what-if questions (Shanthikumar et al., 2007).

4.1 Modeling

Models are used when the real system cannot be engaged because *a*) it may not be accessible, *b*) it may be dangerous to engage the system, *c*) it may be unacceptable to engage the system, or *d*) the system may simply not exist (Banks, 2009).

Banks (2009) and Petty (2009) both stress that a model is an abstraction from something, intended to serve for a specific application and thus providing a suitable description from a certain point of view. Depending on the viewpoint, some characteristics are considered important while others are omitted.

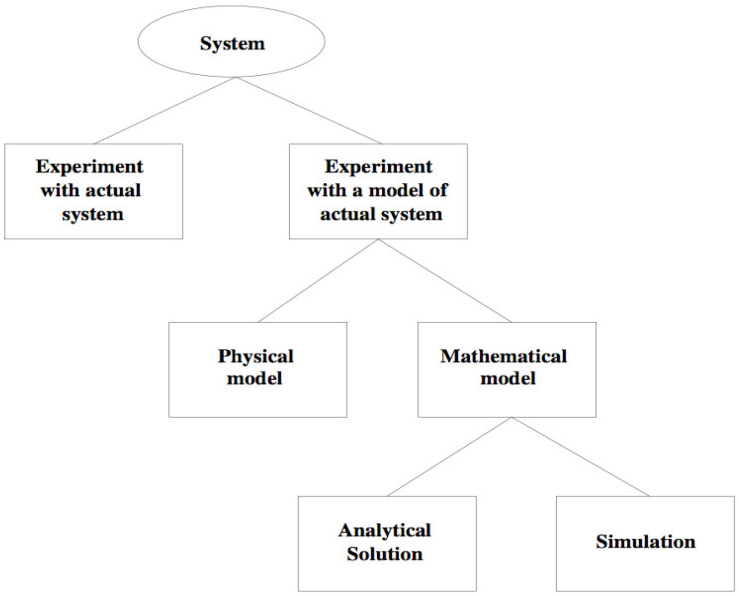


Figure 13: Taxonomy to study a system (Law and Kelton, 2000; Goti, 2010)

Model Typology Petty (2009) broadly groups models into two types: conceptual and executable. Conceptual models document those aspects of the real system that are to be represented and those that are to be omitted. The conceptual model may include mathematical equations, flowcharts, Unified Modeling Language (UML) diagrams, data tables, or simply textual descriptions. The executable model is intended to simulate the real system as specified in the conceptual model. The executable model may be a physical model or a mathematical representation, respectively a computer program. Another classification distinguishes models with respect to techniques used for modeling. Fishwick (1995) proposes four categories: *a*) conceptual modeling, *b*) declarative modeling, *c*) functional modeling, and *d*) spatial modeling.

Mathematical Models Mathematical models can be seen as a set of mathematical equations and logical relationships (Abu-Taieh and El Sheikh, 2008). Among mathematical models, Wainer (2009) further distinguishes between *a*) analytical, *b*) numerical and *c*) simulation approaches. Analytical models provide formal representations that allow us to study the variables of interest in a mathematical system, e.g. a system of differential equations. For models of a higher complexity, respectively for those for which no analytical solution is available, numerical methods are introduced. With respect to continuous problem formulations, respectively those with temporal dimensions, numerical methods employ discretization in order to calculate model variables at predefined time steps. The solution obtained by numerical approximation comes with errors, since it is impossible to calculate every possible combination of the models variables in a reasonable time. In the area of computer simulation, traditional numerical models were converted into computer-based solutions —the model is basically a computer program. Velten (2009) gives an introduction into mathematical modeling and simulation.

4.2 Simulation

Since computer simulation attracts many researchers, numerous definitions have been emerged for the term computer simulation; cf. (Abu-Taieh and El Sheikh, 2008; Paul and Balmer, 1993; Banks, 2009). Despite of the fact that those definitions may differ within a range, most of those definitions put emphasis on the temporal aspect of simulation. Sokolowski (2009) and Petty (2009) for example, simply define simulation as the process of executing a model over time. More precisely, simulation adds a temporal aspect to a static model by depicting how the system being modeled changes over time.

Applications Banks (2009) groups simulation applications into five categories: *a*) training, *b*) decision support, *c*) understanding, *d*) education and learning, and *e*) entertainment.

With respect to the focused use case, Pedrielli et al. (2012) list nine important application areas for simulation: *a*) commerce, *b*) manufacturing, *c*) supply chains, *d*) health services and biomedicine, *e*) simulation in environmental and ecological systems, *f*) city planning and engineering, *g*) aerospace vehicle and air traffic simulation, *h*) business administration and management, and *i*) military applications. Simulation for manufacturing and supply chains is discussed in (Merkuryev, 2009), for health services and biomedicine in (Sokolowski and Banks, 2011), and Adamy (2003) deals with simulation in military applications. In that line Sokolowski and Banks (2009) and Wainer (2009) give numerous examples of simulation models for a wide range of application areas.

4.2.1 Model Typology

Any simulation model is developed for a specific purpose. It shows specific characteristics depending on the targeted system to be modeled and on the chosen simulation method applied. The behavior of a simulation model is generally characterized by the following three aspects: *a*) static or dynamic, *b*) discrete or continuous (in time and/or variables), and *c*) deterministic or stochastic.

Static vs. Dynamic Models Despite of the fact that most definitions for simulation consider the aspect of modeling time as elementary (dynamic models), Goti (2010) states, that M&S also frames models without any components related to time (static models).

Discrete vs. Continuous Models Within the group of dynamic simulation models, Sokolowski (2009) distinguishes between two types of systems: *a*) discrete in which the variables change instantaneously at separate points in time, and *b*) continuous where the state variables change continuously with respect to time. Pedrielli et al. (2012) and Abu-Taieh and El Sheikh (2008) further distinguish between two groups among discrete system simulation systems: *a*) time-based discrete simulation and *b*) Discrete Event Simulation (DES). In time-based discrete simulation (also known as time-slice approach) variables change at predened points in time as the simulation moves forward in equal time intervals. In DES variables change event-based, respectively whenever a new event occurs. In (Abu-Taieh and El Sheikh, 2008) and (Hrúz and Zhou, 2007) one can find a classification scheme with four types of simulation models. This classification scheme emerges when the discrete or continuous nature is determined for time as well as for model variables separately. Then variables in the model change in four ways: *a*) continuously at any point of time (continuous time), *b*) continuously at discrete time events, *c*) discretely at any point of time (continuous time), or *d*) discretely at discrete time events. Wainer (2009) presents a mapping between common M&S techniques and those four groups.

Deterministic vs. Stochastic Models Beside the discrete and continuous nature of model variables and time, there is another feature of importance that characterizes simulation models: the behavior of the system can be deterministic or stochastic. Simulation models may make use of randomness in order to create stochastic effects, which in turn enable the model to imitate system behavior on a certain level of abstraction. Deterministic simulation models always lead to identical results, provided that the simulation system runs under identical conditions. In contrast, the results of stochastic simulations are not exactly predictable in cause of intended random effects that represent a specific behavior (Abu-Taieh and El Sheikh, 2008).

4.2.2 Technologies

Allen (2011) provides an extensive listing of common M&S techniques with additional information, e.g. the basic principle and relevance for certain applications. Zimmermann (2008) emphasizes the importance of Stochastic Discrete Event Systems (SDES), but also mentions popular model classes like Queuing Theory and Petri nets (with stochastic extensions). Queuing Theory employs stochastic processes to model waiting lines under consideration of stochastic effects; cf. (Stewart, 2009; Gautam, 2008; Willig, 1999; Nyhuis and Wiendahl, 2009). Variants of Markov chains are usually used to create queuing models, which may represent manufacturing systems or parts of

it (Xu et al., 2008a). Compared with simulation, analytical approaches based on Littles law and Queuing Theory can be much faster in achieving reasonable results. Littles law states that there exists a proportional relationship between WIP and CT in a queuing system Little (1961). Queuing models provide CT estimations based on the stochastic analysis of the arrival process and the service process (Shanthikumar et al., 2007). Hrúz and Zhou (2007) discuss numerous variants of Petri nets. Another interesting branch in simulation technology is established by agent-based systems, i.e. multi-agent systems. Those systems establish distributed simulation based on communication among autonomous agents; cf. (Jennings and Wooldridge, 1998; Weiss, 1999; Bussmann et al., 2004; Allen, 2011).

Discrete Event Simulation (DES) DES is one of the most important approaches in the area of M&S. The nature of DES systems generally is *a*) dynamic (model varies over time), *b*) discrete (in time and variables), and *c*) stochastic (as it employs randomness). Sokolowski (2009) formally defines DES as the variation in a model caused by a chronological sequence of events acting on it. Events are instantaneous occurrences that may cause variations or changes in the state of a system. The state of a system is defined as one or more variables that completely describe a system at any given moment in time. A system clock keeps track of the simulation time and may be used to trigger events. Allen (2011) draws a bright future for both discrete event simulation and agent-based modeling. He lists four factors that will contribute to their widespread application: *a*) continuing pressures for organizational efficiency, *b*) improved access to low-level data through new sensors and databases, *c*) enhanced visualization capabilities as simulations become more realistic, and *d*) increasing computational efficiencies from faster computers. For deeper insights into DES and for interesting reviews over recent developments in this field, see (Wainer, 2009; Goti, 2010; Wainer and Mosterman, 2011; Allen, 2011; Pedrielli et al., 2012).

4.3 Simulation Project Life Cycles

Any successful simulation project involves skills in both simulation and project management. Robinson and Bhatia (1995) discuss important activities in a simulation project. Structured courses of activities, also known as (simulation) project life cycles, have been proposed for M&S studies in (Balci, 1994, 1998, 2012; Robinson and Bhatia, 1995; Wainer, 2009).

Wainer (2009) summarizes important activities to be performed during a simulation project. For simplification these activities are grouped into five phases: *a*) problem formulation and conceptual modeling, *b*) data collection, *c*) modeling and simulation, *d*) experimentation and output analysis, and *e*) validation and verification. Wainer (2009) further points out that this sequence of steps does not have to be interpreted as strictly sequential. It is highly recommended to follow the proposed steps in a cyclic manner as to achieve incremental developments in each phase.

4.4 Validation and Verification

Wainer (2009) describes Validation and Verification (V&V) as follows: Verification is related to the internal consistency between the conceptual and the executable model, and makes sure that the simulation model is implemented as specified (Did we build the model right?). Validation focusses on the correspondence between model and reality, respectively checks whether simulation results are consistent with the system under study (Did we build the right model?). Refer to Figure 14 for a graphical representation of entities and activities in the area of V&V.

According to Balci (1994), Wainer (2009) and Petty (2009) V&V techniques can be grouped into four categories: *a*) informal, *b*) static, *c*) dynamic, and *d*) formal.

Activities Among others, Petty (2009) describes a very detailed model for the relationships among the entities analyzed or developed during a simulation project. He defines activities to be carried out within the process of V&V. Sargent (2010) proposes a more simple model in which three activities play key roles: *a*) conceptual model validation, *b*) computerized model verification, and *c*) operational validation

Conceptual model validation is determining that the assumptions underlying the conceptual model are correct and the models representation of the problem entity is reasonable for the intended purpose of the model (Sargent, 2010).

Computerized model verification ensures that the implementation of the conceptual model is correct, respectively is conform to the specifications made in the conceptual model. Usually, techniques from software engineering verify that the executable model is implemented as intended, where rudimentary stepwise code debugging would probably one of the simplest forms (Sargent, 2010).

Operational validation is determining whether the simulation model's output has the accuracy required for the model's intended purpose. At the latest in this stage, crucial modeling problems will show up in form of inaccuracies, either related *a*) to invalid data, *b*) implementation errors in the executable model, or *c*) even wrong assumptions in the conceptual model (Sargent, 2010).

Data Validity Invalid data are often reason for failing simulation modeling projects. It is usually difficult, time consuming, and costly to obtain appropriate, accurate, and sufficient data. Data are needed for *a*) building the conceptual model, *b*) validating the model, and *c*) performing experiments with the validated model. In order to ensure high-quality data, one should develop good procedures for *a*) collecting and maintaining data, *b*) testing the collected data using techniques such as internal consistency checks, and *c*) screening the data for outliers and determining if the outliers are correct (Sargent, 2010).

Accreditation Petty (2009) discusses the process of accrediting a model, in addition to the V&V activities: Accreditation is the official certification by a responsible authority that a model is acceptable for use for a specific purpose. The term accreditation is often used in conjunction with verification and validation, even though it is an entirely different sort of process. While verification and validation are technical in nature, accreditation is a nontechnical decision process. Consequently V&V is often extended to Validation, Verification and Accreditation (VVA).

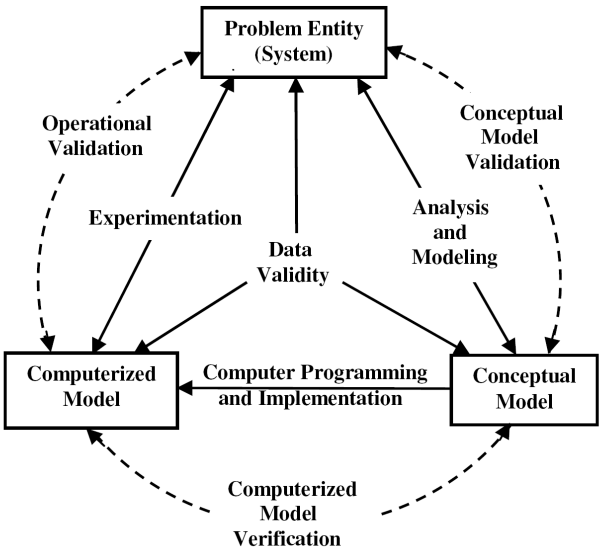


Figure 14: Validation and Verification (Sargent, 2010)

4.5 Input Data Management

Missing data and low data quality is a major barrier in transferring results from academic into real-world applications (Mönch et al., 2011b). Especially the huge amounts of data available in the todays operative systems justify the need for data mining techniques to be used to deal with missing or erroneous data. Sargent (2010) discusses data validity as an integral part of V&V and

emphasizes the importance of appropriate, accurate, and sufficient data for a successful simulation project. Robinson and Bhatia (1995) simply classify data into three categories: *a*) available, *b*) not available but collectable, and *c*) available and not collectable.

Skoogh and Johansson (2008) identify four major problems related to Input Data Management (IDM) during simulation projects; the data problems we face are often due to: *a*) too many measurements in cause of insufficiently specified data accuracy, *b*) late additional rounds of data gathering as a consequence of missing actions verifying that all data would be found, *c*) failing of raw data gathering caused by not properly chosen and clearly defined gathering methods, and *d*) many iterations in data collection as a result of an inefficient validation process.

To overcome these challenges it is highly recommended to develop effective methodologies in the area of IDM, especially in DES that require enormous amounts of data. For example, Bengtsson et al. (2009) describe methodologies for IDM in DES. Another structured methodology for the input data management process that covers identification, collection, and preparation of input data for simulation models is available in (Skoogh and Johansson, 2008). They propose a clear mode of operation for handling input data, intending to increase both the rapidity and the quality in the input data phase of simulation projects. The proposed scheme comprises the following sequence of activities: 1. identify and define relevant parameters, 2. specify accuracy requirements, 3. identify available data, 4. choose methods for gathering of not available data, 5. create data sheets, 6. compile available data, 7. gather not available data, 8. prepare statistical or empirical representation, and 9. validate data representations.

Automated Model Generation The process of data collection is extremely time consuming and hence automating it would be highly advantageous (Robertson and Perera, 2002). Mathewson (1984), probably as one of the first, deals with the idea of an automated model generation, in particular for DES systems. Since then, a considerable amount of literature about challenges and solution approaches related to the idea of automated modeling has been published. Son and Wysk (2001) also present an architecture to automatically generate a simulation model. They describe a methodology to collect static and dynamic information from shop floor control systems, illustrated with examples of different manufacturing systems. In particular data coupling between the data world of the production is considered as a challenging task, and Horn et al. (2005) present a concept to integrate a simulation-based real-time production planning system in wafer fabrication. Especially for online simulation systems, automated generation of simulation models becomes necessary to ensure (near) real-time capabilities. Noack et al. (2010) discuss a data architecture used to automatically create an online simulation model for an entire waferfab. With respect to large-scale simulation models, huge amounts of data from different databases and other data sources need to be processed properly; cf. (Randell and Bolmsjö, 2001; Mueller et al., 2007).

Data Cleaning The success of automated modeling is based on valid data. Though entirely valid data are rarely the case, automated modeling is strongly related to data cleaning procedures.

Rahm and Hai Do (2000) classify data quality problems that are addressed by data cleaning and provide an overview of the main solution approaches. Data cleaning is especially required when integrating heterogeneous data sources. In particular, simulation projects for complex manufacturing facilities (e.g. waferfabs) typically deal with numerous, heterogeneous data schemes. Schema matching is the task to produce a mapping between elements of minimum two data schemes that might be heterogeneous in their structures, and is a basic problem in many database application domains Rahm and Bernstein (2001), to which large-scale simulation projects definitely belong to.

Fang et al. (1991), Bright et al. (1994) and Rahm and Bernstein (2001) discuss strategies to tackle heterogeneity in connected databases.

4.6 Simulation in Waferfabs

Semiconductor manufacturers are constantly under pressure to reduce CT and OTD. Accurate CT estimation can greatly support production planning and scheduling of waferfabs. However, this question is not easy to answer due to complicated tool specifications and process flows (Shanthikumar et al., 2007).

Fowler and Rose (2004) state that there is a need for the pervasive use of M&S for decision support in current and future manufacturing systems. They identify four grand challenges that need to be addressed by the simulation community to realize this vision; M&S particularly needs: *a*) an order of magnitude reduction in problem-solving cycles, *b*) real-time, simulation-based problem-solving capability, *c*) true plug-and-play interoperability of simulations and supporting software, and *d*) to convince the management to sponsor M&S projects. Between those four convincing the management to sponsor modeling and simulation projects is the biggest challenge.

In addition, grand challenges in M&S are discussed in (Fowler and Rose, 2004) for complex manufacturing systems and in (Mönch et al., 2011b) for discrete event logistics systems. Uzsoy et al. (1992b) give a review of simulation performance evaluation in semiconductor industry. Chien et al. (2008) review the role of modeling and analysis in semiconductor manufacturing, presenting expert's views on the challenges and successes of modeling and analysis.

Wafer fabrication poses challenging difficulties in developing simulation models, in particular due to two reasons: complicated tool specifications and varying process flows (Shanthikumar et al., 2007). First, service processes on tool sets are subject to high variation due to *a*) multiple products and operations on the same tool set, *b*) requirements of cascading and setups, *c*) restrictions such as dedication and waiting for metrology verification, *d*) variable batch sizes, and *e*) scheduled and non-scheduled downtimes of equipment.

Second, process flows involve hundreds of operations with many reentrant processes. Thus, process flows, even for identical products, vary due to *a*) unforeseeable effects like scrap, on-hold, and rework, *b*) multiple products accompanied with lot split and lot merge, *c*) engineering lots that compete with production lots for resources (in R&D waferfabs), and *d*) random order arrivals and product replacements as a result of fast market demand changes.

Queuing Theory vs. DES In accordance to the two basic approaches to study a model of a system, analytical modeling and simulation, there exist two prevailing methods for waferfab performance evaluation: Queuing Theory and DES. Both approaches have their justification, are extensively used in practice and complement each other (Uzsoy et al., 1992b). A decision for or against one of both is recommended to be driven by the profile of requirements given by the use case(s).

Analytical modeling may be a better choice for simple queuing systems, whereas simulation is often used to analyze the complex queuing systems in which analytical methods become intractable or unacceptable due to too inaccurate outputs (Fishwick and Park, 2009). Queuing models are used for fast, approximate analyses and simulation models are developed for detailed studies which take considerably longer (Uzsoy et al., 1992b); cf. (Shanthikumar et al., 2007).

Uzsoy et al. (1992b) report that the values of the parameters of interest obtained from the queuing models deviated from the values obtained from simulation by between 7% and 20%. However, the run times observed for the queuing approach were much shorter than the simulation run times.

Shanthikumar et al. (2007) proceed with two aspects that make exploring what-if question by simulation difficult and extremely time-consuming: First, simulation requires an enormous amount of input data and substantial resources to maintain and update the model. Second,based on the nature of simulation modeling, multiple replications are needed to perform confident statistical analysis.

Likewise, one can find arguments against applying Queuing Theory. One of the Queuing Theory's major issues in practical applications is: Queuing Theory assumes a stationary status of the system and real fab operation is never in a stationary status (steady-state), but at any time in a transient state (Shanthikumar et al., 2007). Shanthikumar et al. (2007) survey the application of Queuing Theory for waferfabs and Govil and Fu (1999) survey the contributions and applications of Queuing Theory in the field of discrete part manufacturing. Uzsoy et al. (1992b) provide an extensive list of simulation models applied in semiconductor manufacturing and also present examples of queuing models. Kumar and Kumar (2000) provide an instruction to the application of queuing models to the design and analysis of waferfabs.

4.6.1 Forecasting

Forecasting generates expectations of the future in order to evaluate alternate policies (Hopp and Spearman, 2001). The development of advanced forecasting techniques is an important aspect in the area of modeling and analysis in semiconductor manufacturing (Chien et al., 2008). There is one basic distinction between forecasting methods: *a*) qualitative forecasting using the expertise of people, rather than precise mathematical models, and *b*) quantitative forecasting based on some kind of mathematical model (Hopp and Spearman, 2001). Among quantitative forecasting models, Hopp and Spearman (2001) further distinguish between: *a*) causal models predicting a future parameter as a function of other parameters, and *b*) time series models predicting a future parameter as a function of past values of that parameter. A commonly used classification scheme distinguishes forecast methods with respect to their forecast horizon: *a*) long-term forecasts to support strategic decisions (months to years), *b*) mid-term forecasts to support tactical decisions (weeks to months), and *c*) short-term forecasts to support operational decisions (days to weeks).

In that context, Hopp and Spearman (2001) state the following three well-known and generally applicable laws of forecasting: *a*) Forecasts are always wrong! *b*) Detailed forecasts are worse than aggregate forecasts! *c*) The further into the future, the less reliable the forecast will be!

Long-Term Forecasting Long-term forecasting is usually performed as offline study, and the simulation experiments are conducted as steady-state simulation. Uzsoy et al. (1992b) provide an extensive list of simulation models applied in semiconductor manufacturing. Simple factory models show essentially the same behavior as a complete factory (Rose, 1999b). Rose (2000) investigates model accuracy of simple waferfab models and proposes model improvements in (Rose, 2007). Long-term forecasts are used to predict up to two years of fab operations (Bosch and Wright, 2008). Gißrau (2013) presents a DES system developed for a highly customer oriented ASIC factory. In order to improve model accuracy, researchers investigate specific topics in connection with waferfab simulation modeling. For example, modeling machine breakdowns in waferfab models is an important task, discussed in (Rose, 2004; Scholl, 2008)

Long-term forecasting is a suitable use case for analytical approaches, e.g. Queuing Theory (see Section 4.6). For example, Schelasin (2011) outlines a method based on Queuing Theory together with targeted historical data to estimate CT. Becker (2003) presents a simulation model of a complete waferfab using Petri nets. A promising branch in forecasting technologies employs data mining techniques. Based on measured and calculated process metrics (such as WIP at specific operations, lot priority, product type, etc.), data mining algorithms feed models used to predict, e.g. CT (Backus et al., 2006). Predictions based on data mining algorithms perform better when being combined with domain knowledge (Chien et al., 2005). Another idea to enhance forecasting schemes based on data mining methods, is to implement clustering methods, which support the system to specifically threat product groups, tool sets, process types etc. according to their specifics. This way, Mosinski et al. (2011) implement a lot delivery forecast based on time series in a waferfab with wide product range. Neural networks have also been investigated. Usually back-propagation networks are trained with historical data to generate forecasts (Yu and Huang, 2002; Chen, 2007). Based on neural networks, Liao and Wang (2004) generate delivery time estimates for 300 mm AMHS operations.

Short-Term Forecasting For short-term forecasting, DES is the option of choice, since analytical approaches based on Queuing Theory lack the required level of detail; cf. Section 4.6. Simulating short terms is naturally characterized by a transient behavior. In most cases, it is desired to setup those systems as online implementations, which continuously synchronize with the activities on the shop floor in order to instantly provide short-term forecasts. Weigert et al. (1999) investigate the basic principles of process accompanying simulation. They discuss a method for synchronization and adaptation of the simulation model while the manufacturing process continues. High fidelity simulation systems that provide reliable short-term forecasts are highly desired to be applied in operational planning, scheduling, and control of manufacturing; cf. Smith et al. (1994); Reijers and van der Aalst (1999); Werner and Weigert (2002); Bagchi et al. (2008). The existence of actual data is essential to any short-term simulation system. Online simulation systems have direct access to the current fab state and and thus provide the capability to generate predictions in

(near) real-time; cf. Drake and Smith (1996); Potoradi et al. (2002). High fidelity combined with the ability for fast responses, makes online short-term forecasting highly beneficial for operational planning, scheduling, and control of manufacturing systems But, implementing such a simulation system is a crucial task. Scholl (2008); Scholl et al. (2010, 2011); Noack et al. (2011) describe an online simulation model for short-term lot arrival forecasting in a mature 200 mm waferfab. They discuss challenges, solution approaches and important modeling issues; cf. Noack (2012).

4.6.2 What-If Studies

Considerable effort has gone into the development of simulation models for waferfabs and their use to analyze the effects of different control strategies and equipment configurations (Uzsoy et al., 1992b). Simulation models also serve as vehicle to investigate product mix changes, fabrication layouts, and the effect of lot sizes. Simulation studies that are carried out to answer certain questions in order to facilitate understanding of the relation between cause and effect, are also known as what-if-studies.

Dispatching Rule Evaluation Rule-based dispatching rules constitute one of the most powerful tools to control material flows. Early simulation studies on how certain dispatching rules affect factory performance are presented in (Uzsoy et al., 1992a; Waikar et al., 1995; Holthaus and Rajendran, 1997; Mittler and Schoemig, 1999). The effect of various dispatching rules on CT and OTD have been assessed by use of DES systems for example in (Kim et al., 2001; Rose, 2001, 2002, 2003a,b; Hung and Chang, 2002; Sunkara and Rao, 2003; Dominic et al., 2004; Mönch and Zimmermann, 2004). More recent DES studies about dispatching control strategies are available in (Sha et al., 2006; Valente, 2007; Ko et al., 2010; Chiang and Fu, 2012; Gißrau, 2013; Zhou and Rose, 2009, 2010, 2012).

Layout Studies Layout studies are considered as an important activity in the waferfab planning phase, motivated by the fact that the layout is difficult and expensive to modify once it is set up (Chung and Jang, 2007). Hase et al. (1994, 1997) and Chang and Chang (1998) present DES studies in which they study cellular, reentrant, layer-based layouts and variants of them for waferfabs. In (Campbell and Ammenheuser, 2000) and (Quinn and Bass, 1999) one can find factory layout studies accompanied with AMHS modeling for modern 300 mm waferfabs. More recent studies on issues related to factory layout and automation are presented in (El-Kilany, 2004) and (Chung and Jang, 2007).

Lot Release Strategies Lot release strategies intend to keep the WIP in waferfabs under a critical level and thus prevent the increase in CT; cf. Little's Law (Little, 1961). Popular workload control strategies are presented in (Rose, 1999a) and (Sivakumar et al., 2008). Recently, Fredendall et al. (2010) investigate the effect of lot release strategies via simulation.

Lot-Sizing Lot sizing is considered to be a promising strategy for CT reduction, especially for high-mix, low-volume factories. Combining Queuing Theory and simulation together as one unified approach, Potoradi and Winz (1999) make a general observation concerning lot sizes. For areas of the factory that are highly utilized, a larger lot size is required to meet THP. For areas less utilized, a smaller lot size can be implemented to minimize CT. Since the process time depends on the lot size for a wide range of machine types, it is beneficial to match the lot size and machine configuration (Schmidt et al., 2006). The effect of matching lot sizes and machine configurations considerably depends on loading situations at factory bottlenecks (Wang and Wang, 2007). Stubbe (2010) examines the effect of small lot sizes on CT in combination with a conversion of batch processes to mini-batch or single-wafer processes. Especially for batch processes, lot sizing plays a key role in performance evaluation. One can find related simulation studies in (Rummel, 2000; Bonnin et al., 2003).

4.6.3 Simulation-Based Scheduling

Simulation-based scheduling systems place highest requirements upon simulation. Scheduling systems require an online simulation that provides detailed forecasts with high accuracy in a very short time. Qiao et al. (2012b) review simulation-based scheduling approaches for wafer fabrication and introduce a simulation-based modular planning and scheduling system for a waferfab. Smith et al. (1994), Drake and Smith (1996) and Sivakumar (1999) present early scheduling systems based on DES in complex manufacturing environments, e.g. wafer fabrication. More simulation-based scheduling systems used for shop floor planning, control and scheduling are presented in (Werner and Weigert, 2002; Potoradi et al., 2002; Chong et al., 2003b; Mönch et al., 2003; Klemmt, 2012). Chong et al. (2003a) and Fowler et al. (2003) describe simulation-based scheduling approaches that focus on factory bottlenecks. Horn et al. (2006), Horn (2008) and Weigert et al. (2009) discuss their experiences during the development of a simulation-based scheduling system for a semiconductor backend facility. A simulation-assisted approach for scheduling and rescheduling complex production system configurations is investigated in (Dangelmaier et al., 2006, 2007). Wu and Wysk (1989) and Zhang et al. (2009a) describe their strategies to dynamically choose suitable dispatching rules based on the results provided by DES.

4.7 Wafer Fabrication Equipment Modeling

Equipment models form one of the central pillars to evaluate waferfab performance (planning and simulation) and to control material flow (sophisticated dispatching and scheduling in particular). Generally it is distinguished between two types of equipment models: *a*) simplified equipment models based on an analytical approach that is sufficiently fast enough to be implemented as a basic component in a simulation/dispatching/scheduling system, and *b*) detailed simulation models used for performance analysis, in most cases applied to study complex cluster tools.

Since detailed simulation models perform too slow to be applied for fab simulation, most research activities try to develop fast, analytical models with higher accuracy. Conventional equipment models used for long-term simulation applications usually consist of information about THP and may include additional information for BP. Simulation especially for short horizons and optimization solutions (e.g. lot scheduling), need more detailed models that mimic the equipments processing behavior more precisely. Beyond rather simple THP models, capacity equipment limitations as well as predicting accurate processing times become points of interest in the field of equipment modeling. The demand for detailed equipment models separately focusing on capacity-related and temporal aspects of lot processing is growing, driven by upcoming simulation and optimization applications.

Automated Parametrization The need for more detailed models with high accuracy remarkably increases the modeling effort. Building equipment models is usually done by hand and takes large amounts of time due to manually searching multiple data sources and analyzing surveys addressed to experts. In spite of the fact that modern MESs track huge amounts of data generated on the shop floor, not all needed modeling information are directly accessible. Compared to expert interviews, the idea of automated modeling based on given MES data promises considerable time savings and even new information that can increase equipment model quality. Self-creating and self-parameterizing equipment models as a result of automating the modeling process using data mining techniques as far as possible would considerably increase efficiency of today's modeling policies.

4.7.1 Analytical Models

Any waferfab simulation system incorporates a kind of analytical model. Shikalgar et al. (2003) discuss a realistic way of representing cluster tools in a simulation model of the entire line. Most publications focus on modeling modern cluster tools, because of their modeling complexity and growing importance in industry. Morrison (2011a,b) discusses linear and affine models that are commonly used to model equipment THP in waferfab simulations. He focuses on clustered photolithography and multi-cluster tools and develops flow line models that allow for diverse products, wafer lots and wafer location dependent setups. The processes in photolithography are often performed by linear cluster tools, Yi et al. (2007) analyze steady-state THP and scheduling

for those with with single-blade robots. Analytical models need to consider the effect of parallel chambers and their interrelation with small lot sizes, having that focus. Schmidt et al. (2006) evaluate modeling methods for small lot sizes for cluster tools with parallel chambers.

Wood et al. (1994); Wood (1996) lay the basis for analytical cluster tool models. They introduce an analytical modeling approach used for THP modeling and process time estimations. It is used for cluster tools that include internal wafer handling and processing times and additionally provides bottleneck conditions for serial and parallel processing modes.

Perkinson et al. (1994, 1996) present an analysis of the relationship between process times, transport times, and maximum THP in an individual cluster tool, discussing the effect of redundant chambers and chamber revisitation process sequences on the THP. Gupta et al. (2008) verify the models described in (Perkinson et al., 1994, 1996) by use of simulation.

In the field of automated modeling, Lange et al. (2008) present an approach for automated generation of equipment THP models by analyzing internal equipment events, basically combining the approaches of (Wood et al., 1994; Wood, 1996) and (Perkinson et al., 1994, 1996); cf. (Lange, 2008). A method for automated semiconductor equipment modeling and model parameter estimation using MES data is presented in (Kohn et al., 2010; Kohn and Rose, 2011). Frantsuzov (2011) validates automatically created equipment models, their applicability and accuracy, using various sets of real-life data from wafer fabrication. Also based on event data, Hosoe et al. (2007) investigate estimating tool processing time with high accuracy.

Niedermayer and Rose (2003) observe the influence of recipe combinations and the impact of start delays and present the idea of using slow down factors mirroring dynamic interrelationships inside a cluster tool. In the following, Unbehaun and Rose (2006) have continued developing the idea (of using slow down factors) and present a model as well as a method to predict process times at cluster tools. Niedermayer and Rose (2003) and Unbehaun and Rose (2006) both evaluate their results by use of DES models mirroring the ideal equipment behavior in real world. With respect to real-world data, Kohn and Rose (2011) present an approach to automatically create an analytical process time model. They consider the effect of small lot size as well as the slow down effect. Other simulaton models employ regression spline meta-models (Ruppert et al., 2000) or Petri nets (Qiao et al., 2012a).

4.7.2 Simulation Models

Cluster tools are usually subject to simulation models, since analytical models lack of accuracy due to the highly complex internal processing behavior. Early simulation models for cluster tools are discussed in (Pierce and Drevna, 1992; LeBaron and Pool, 1994). More sophisticated models reproduce the flow of wafers through a cluster tool more accurately (Becker, 2007). For example Park and Morrison (2011) develop a simulation of cluster tools with realistic parameters, which incorporates rolling setups and wet cleans. LeBaron and Hendrickson (2000) present a flexible and sufficiently accurate cluster tool simulation model. Simulation models support cluster tool performance evaluation in order to provide reliable THP values for factory capacity planning. Koehler et al. (1999) describe the application of simulation for analyzing cluster tool CTs and cluster tool capacity planning.

Another use case is driven by industrial engineering, simulation models facilitate identifying internal bottlenecks, e.g. a slow moving wafer handler. Once a bottleneck is identified, it is possible to initiate appropriate measures that improve cluster tool's THP. Swe et al. (2006) present a simulation model for cluster tools. They discuss the factors that influence CT. Christopher (2008) shows the effect of load lock dedication on a sample multi-process chamber tool.

A further application is given by the need to validate analytical models —Gupta et al. (2008) verify the models described in (Perkinson et al., 1994, 1996) by use of simulation. Detailed simulation models also serve as a suitable vehicle to evaluate dispatching and scheduling strategies for cluster tools. For example, Dümmler (2004) deals with modeling and optimization of cluster tools in semiconductor manufacturing. The use of slow down factors and their application to cluster tool scheduling is discussed in (Niedermayer and Rose, 2003; Unbehaun and Rose, 2006, 2007). Various scheduling methods are subject to simulation studies. Oechsner and Rose (2005) deal with filtered beam search and recipe comparison and Jung and Lee (2012) employ timed Petri nets.

4.7.3 Modeling Equipment Capacity (External Behavior)

Kohn et al. (2010) present a system of quantities that classifies capacity-related equipment model parameters according to their relationship to three dimensions and three units. These dimensions and units refer to physical components of the production system as well as to logical entities being part of the material flow control system. A set of capacity-related equipment parameters is arranged into an ordinal system with the goal to create a helping structure and a better understanding for shop floor manufacturing operations.

On one hand, three unit types build an ordinal scale of this system of quantities when ordering them by their logical relationships, namely *a*) wafer, *b*) lot, and *c*) batch. The lowest unit in a frontend semiconductor fabrication facility and also in this system of quantities is defined as a single silicon substrate, termed as a wafer in this paper. A single lot constitutes the next greater unit within this ordinal system and is defined as a group of one or more wafers of the same type. Obviously both units, wafer and lot, have a physical context in this system. The largest unit is defined by a single batch and has a logical meaning in contrast to the both units mentioned before. A (parallel) batch is commonly defined as a group of lots to be simultaneously processed on a machine, e.g. CFPs such as vertical and horizontal furnaces. Due to the fact that the lots in a batch collectively share the same process resource, the lots have equal starting and finishing times.

On the other hand, three dimensions describe the size of a physical or logical entity with a capacity-related meaning, namely *a*) lot size, *b*) batch size, and *c*) equipment size. First, the size of a lot, given in the number of wafers, is commonly known as lot size and identically termed in this system. Second, the common term batch size is used to describe the size of a batch and can be set either in the number of lots or wafers. The last dimension, equipment size, describes the equipment's process capacity. The process capacity is defined as the maximum number of units that can be processed simultaneously throughout the equipment and can be expressed in units of batch, lot and wafer.

As a result of these presented dimensions and units, six meaningful capacity-related model parameters emerge: *a*) lot size in wafers (LSW), *b*) batch size in lots (BSL), *c*) batch size in wafers (BSW), *d*) equipment size in batches (ESB), *e*) equipment size in lots (ESL), and *f*) equipment size in wafers (ESW). The dimension lot size can only be meaningfully described with the number of wafers within the lot carrier (LSW). The dimension batch size is usually defined by a maximum number of lots (BSL). In some cases the compilation of lots to a batch is additionally limited by the sum of the lots wafers (BSW). The dimension equipment size can either be limited by the number of simultaneously processed batches (ESB), the number of simultaneously processed lots (ESL), the number of wafers that could be processed in parallel (ESW) or by a combination of all of them. In some cases there exists a mathematical relationship between the values of these model parameters. One can see that this system shows an ordinal character for both dimensions that combine physical and logistic entities of the production system.

Refer to Figure 15 for a graphical representation of the described capacity model.

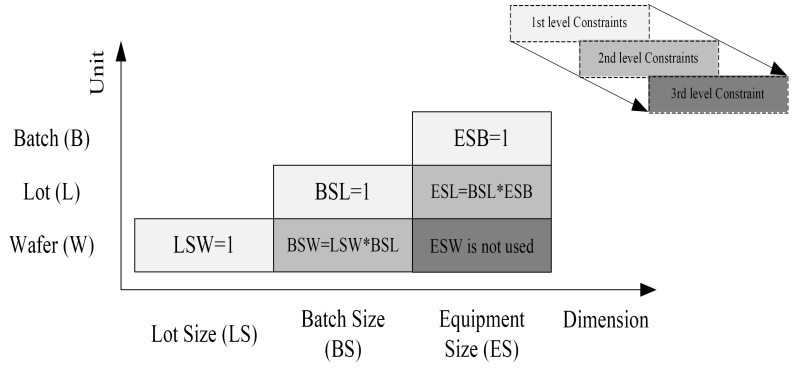


Figure 15: System of capacity-related parameters to model wafer fabrication equipment (Kohn et al., 2010)

4.7.4 Modeling Processing Time (Internal Behavior)

A common approach to classify wafer fabrication equipment is to distinguish equipment with respect to their processing time behavior, respectively the relationship between lot size and process time. Among others, Schmidt et al. (2006) describe three semiconductor tool types. Hosoe et al. (2007) extend these three types by one another special form of SWP. In (Scholl et al., 2010) one can find five types of equipment. They furthermore distinguish between different cluster tools. Schmidt et al. (2006) evaluate modeling methods for small lot sizes for cluster tools with parallel chambers.

Here it is basically distinguished between: *a*) Single wafer tools process single wafers; the process time is a linear function of the lot size. *b*) Batch tools process batches of one or multiple lots; the process time is a constant exclusively depending on the type of process (recipe). *c*) X-piece tools (mini-batch-tool) process batches of x wafers; the lot size is smaller than the standard lot size and the process time is a step function of lot size. *d*) Integrated processing where cluster tools process multiple lots in parallel, sequential or mixed mode; the process time is hard to predict.

Refer to Figure 16 for a graphical representation of the described processing time models.

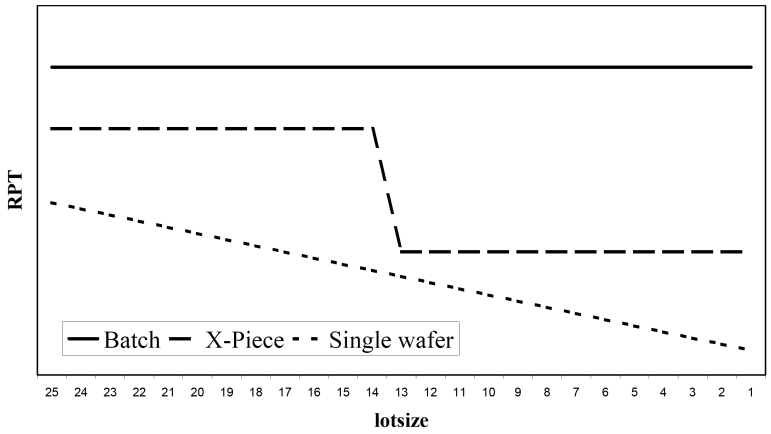


Figure 16: Raw process time (RPT) of different tool types dependent on lot size Schmidt et al. (2006)

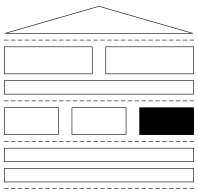
Single-Wafer Processing (SWP) Typical SWP equipment is characterized by a linear relationship between the process time and the lot size. A simple $Ax + B$ model sufficiently predicts process time for a certain recipe, where x denotes the lot size; the slope of the line A and the offset B depend on internal equipment configuration, e.g. chamber process speeds. In most cases, the machine only contains one single process chamber that is capable of processing one single wafer at a time. Consequently, the wafers of a job are processed sequentially and thus the lot size affects the process time directly. The data analysis shows that processing time additionally depends on the job recipe. Moslehi et al. (1992) present an overview of various single-wafer integrated semiconductor device processing.

Batch Processing (BP) BPMs are characterized by a process time that is equal for all the jobs that constitute the batch. The process time minimum depends on the recipe chosen for the process, but is independent from the lot size. A typical batch process in wafer fabrication is performed in CFP machines, i.e. vertical and horizontal furnaces. But also wet benches provide batching capability, processing up to two lots in a batch simultaneously.

X-Piece Processing The operation of X-piece processing machines is similar to the operation of BPMs. They are alternatively called mini-batch equipment. This type of equipment is characterized by a step-function between process time and lot size, whereas the equipment only processes one single job at a time. Internally, the equipment performs BP with wafers. A bunch of wafers from a single job is internally processed simultaneously and these mini-batches of wafers are processed in sequence. The behavior of a step-function arises in case the lot size is greater than the mini-batch

size of the machine, then it takes several mini-batch cycles to finish processing an entire job. For this kind, equipment for ion implantation is representative, where up to 13 wafers are exposed to ion beams on a rotating disc; consequently, it requires two mini-batch cycles to process a regular lot with 25 wafers.

Integrated Processing Especially modeling and simulation of integrated processing equipment (cluster tools) is a crucial task; cf. (Mönch et al., 2011a). The processing behavior is hard to predict due to a complex arrangement of interacting wafer handlers and process chambers, controlled by internal dispatching or scheduling systems. Early configurations comprise one mainframe connected to a number of process chambers, which are extended to architectures that combine two mainframes with numerous process chambers. Most recent cluster tool designs lead to linear cluster tools that promise to be advantageous in THP and CT over their predecessors. For the simplest cluster tool architecture with one mainframe, two load-locks and a number of process chambers two processing schemes stand out, sequential and parallel processing. A cluster tool may switch between both processing modes with the recipe(s) chosen to process the jobs. On one hand, the parallel processing mode in which simultaneously processed lots slow each other down. The internal scheduler threats the lots equally, the lots compete for internal resources and thus slow each other down. On the other hand, the sequential processing mode in which the succeeding lot waits for the preceding lot to finish. Simply, the preceding lot is always preferred by the scheduler, while the secondly started lot is strictly forced to wait.



5 Metaheuristic Optimization

Contents

Optimization is the process of selecting the best alternative among a given set of options. It covers scientific methods for decision making in order to optimize one or more objectives in a constrained environment (Baghel et al., 2012). The science branch Operations Research (OR) is concerned with methodologies for optimization, respectively decision making. Most of the definitions of OR emphasize that OR problems have an interdisciplinary character and that OR methods make use of mathematical models to facilitate the process of decision making (Ravindran, 2008). At this point this work refers to (Hillier and Lieberman, 2001) for a comprising introduction to OR, to (Ravindran, 2008) for a more detailed view on OR with respect to management science and to (Russell and Norvig, 1995) for a very detailed description of methods in the wide area of Artificial Intelligence (AI) that is naturally connected to OR. In the OR community metaheuristics play an increasingly important role in the area of methods discussed for solving optimization problems.

Project Guidelines Talbi (2009) emphasizes that decision making must be tackled in a rational way and describes four basic steps that need to be processed during an OR study: *a*) formulate the problem, *b*) model the problem, *c*) optimize the problem, and *d*) implement a solution (see Figure 17). By formulating the problem the internal and external factors and the objective(s) are outlined. Then, during the modeling phase, an abstract mathematical model is built for the problem, involving simplifications and approximations in order to reduce the complexity. After modeling the problem a suitable method for solving it hast to be found. Finally, the obtained solution is implemented and tested by practitioners in the real environment. In a line with the life cycles in software development and simulation studies, the course of actions in OR projects is rarely linear, but often cyclic (Talbi, 2009). At this point, this work refers to (Hillier and Lieberman, 2001) for a similar OR project guideline, comprising six phases.

Especially for metaheuristics, Hansen and Mladenović (2003) propose four important activities to provide the underpinnings before developing a solution based on metaheuristics for a given problem: *a*) evaluate the difficulty/complexity of the problem and, if possible, the complexity of the best-known exact algorithm, *b*) evaluate the performance of previous algorithms and determine the largest instances solved exactly, *c*) evaluate the performance of previous metaheuristics applied to this problem in terms of size, error and computing time, and *d*) analyze already proposed metaheuristics and identify algorithmic key ingredients used for search, e.g. neighborhoods; cf. (Talbi, 2009).

Optimization Problem The term problem generally refers to a task or question, defining the environment in which a decision has to be chosen, usually with unspecified values. In contrast, the term (problem) instance refers to a certain variant of a problem with specified values (Dorigo and Stützle, 2004).

In the following a formal description of optimization problems is given, corresponding to the description given in (Baghel et al., 2012); cf. (Talbi, 2009). An optimization problem $P = (S, f)$ consists of two components linked with each other: the state space S and the objective function f. The state space S, alternatively referred to as search space or solution space, contains the entire set of feasible solutions, where each solution s satisfies the constraints. The state space S is

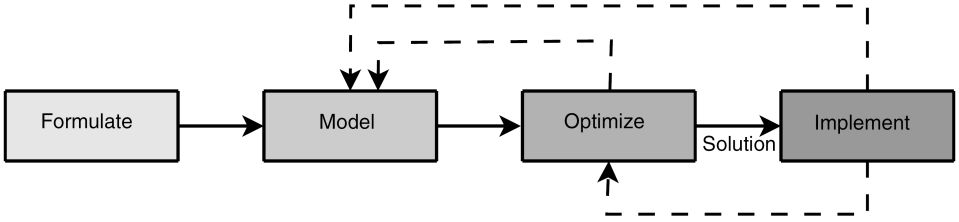


Figure 17: Guidelines for optimization projects (Talbi, 2009)

defined by a set of variables $X = \{x_1, \ldots, x_n\}$, their corresponding variable domains $D_1, \ldots, D_n$, and a set of constraints among the variables in X; $S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n)\} \mid v_i \in D_i\}$. The objective function $f : S \to \mathbb{R}$ assigns to every feasible solution $s \in S$ an objective value $f(s)$ indicating the quality of the solution, formally defined as a real number. Given the fact that we usually focus on minimization problems, the problem is to find a solution $s^* \in S$ with the smallest possible objective value $f(s^*)$. The solution $s*$ is then considered as the best solution, respectively the optimal solution or global optimum; $f(s^*) \leq f(s) \forall s \in S$.

Combinatorial Optimization Similar to simulations models, the optimization problems we face in the area of OR can either be programmed with real valued variables or with discrete variables. Those entirely based on discrete variables belong to the class of COPs. Solving a COP means to choose the best solution from a finite set of possible solutions (Baghel et al., 2012).

The TSP and the Knapsack problem are probably the most common representatives in the class of COPs. The TSP is the problem of finding a minimum length circuit of a graph, where each node of the graph is only visited once (Hamiltonian circuit). The Knapsack problem is the task to select a subset of items from a given set of items, each assigned with a value and a resource requirement, in such a way that they fit into a knapsack of limited capacity, while the sum of item values is maximized (Dorigo and Stützle, 2004). For more detailed information about problems and methods in the area of combinatorial optimization, see (Lee, 2004), (Du and Pardalos, 2005), (Vasudev, 2007), (Paschos, 2008), and (Korte and Vygen, 2012).

Stochastic Combinatorial Optimization Another distinction is given by the existence of uncertainty or stochastic effects in the optimization model: optimization problems can either be deterministic or stochastic. Stochastic Combinatorial Optimization Problems (SCOPs) include uncertain, stochastic, and dynamic information in their mathematical formulations. Most real-world problems come with uncertainties and information about the problem is partially unknown. The lack of detailed information is responded by modelers with assumptions on probability distributions that describe parts of the problem stochastically. In consequence, the optimization problems become even more difficult (Bianchi et al., 2009).

Shortcomings of Exact Methods The most obvious idea to solve a COP is to just enumerate all feasible solutions. But due to the complexity of combinatorial problems, simple complete enumeration will result in too long computing times that are not acceptable in practice. The challenge is to develop efficient algorithms that perform better than simple enumeration (Lee, 2004).

For a considerable amount of optimization problems present in academia and industry, it is intractable to obtain optimal solutions by the use of exact methods in a reasonable time. The crucial point is that exact methods need large amounts of time to solve that kind of problems to optimality. Consequently, the use of exact methods becomes inapplicable for most practical applications, where a responsible person has to make a decision as soon as possible in order to achieve desirable results (Marti and Reinelt, 2011). The reason why problems are hard to solve can be found in their complexity, their size, their specific structure, or a combination of all aspects; cf. (Talbi, 2009).

Advantages of Metaheuristics In contrast, metaheuristics lead to acceptable solutions in a reasonable time; solution quality and computing time is generally not exactly defined, i.e. acceptable and reasonable. In this context, the specific use case defines the range of acceptable solutions and determines up to which deadline computing times are still reasonable. But, a crucial issue is that metaheuristics do not provide the capability to evaluate the solution quality with respect to optimality. Compared to exact optimization algorithms that guarantee the optimality of the obtained solutions and to approximation algorithms that provide at least a value for the distance to the optimum metaheuristics generally lack performance in that point (Talbi, 2009). Metaheuristics primarily justify their use with a well-balanced performance characteristic that describes a favorable trade-off between solution quality and computing time.

Another point that justifies the use of metaheuristics instead of exact methods is given by optimization problems that deal with uncertainty and stochastic effects (SCOPs). For that class of noisy problems, uncertainty and robustness cannot be modeled analytically and thus non-deterministic optimization models are used (Talbi, 2009). In consequence SCOPs become even more difficult, exact methods become inefficient with respect to computing time and metaheuristics emerge as the more attractive alternative, especially for SCOPs (Bianchi et al., 2009).

Further References For a more comprehensive and detailed view on metaheuristics, this work refers to (Blum and Roli, 2003), (Gendreau and Potvin, 2009), (Luke, 2009), (Moscato and Cotta, 2009), (Talbi, 2009), and (Zäpfel et al., 2010), whereas (Talbi, 2009) deserves special mention. Khajehzadeh et al. (2011), Parejo et al. (2011), Baghel et al. (2012), and Boussaïd et al. (2013) present surveys reviewing recent developments in the field of metaheuristic optimization methods.

5.1 Taxonomy

This section briefly outlines the methods available for solving COPs/SCOPs, while following a taxonomy identical to the one presented in Talbi (2009). See Figure 18 for a graphical visualization of the taxonomy described in the following.

Exact vs. Approximate Methods There exist two fundamentally different classes of methods available to solve combinatorial problems: exact and approximate methods. Exact algorithms are characterized by the ability to proof the optimality of the obtained solutions; and by definition this is what distinguishes them from approximate methods (Dorigo and Stützle, 2004; Talbi, 2009). In this context, Prestwich (2008) refers to complete and incomplete search, which correspond to exact and approximate methods, emphasizing that exact methods completely search the state space while approximate methods only search parts of it.

By not having the burden to proof the optimality, approximate methods leave parts of the state space unvisited and thus lead to near-optimal solutions in a reasonable time compared to exact algorithms. Especially for $\mathcal{NP}$-hard problems exact algorithms perform poor with respect to computing time. Consequently solving large instances with exact methods is practical impossible, i.e. would take enormous amounts of time to obtain the optimal solution. Approximate algorithms trade optimality for efficiency (Dorigo and Stützle, 2004); cf. (Talbi, 2009). Obviously complete and incomplete search have complementary strengths and weaknesses (Prestwich, 2008).

Heuristics vs. Approximation Algorithms Approximate methods can be further divided into heuristics and approximation algorithms. A heuristic is any approach without a formal guarantee of performance. Approximation algorithms guarantee that the obtained solution lies within a defined range of the global optimum (Brucker, 2007; Talbi, 2009).

Constructive Heuristics vs. Search Heuristics Within the class of heuristics, it is basically distinguished between constructive heuristics and search heuristics (Zäpfel et al., 2010); cf. (Talbi, 2009). Construction algorithms describe an incremental procedure: starting from an empty initial solution, they iteratively add solution components until a complete solution is obtained without any backtracking. In its simplest version the solution components are added in a random order. More sophisticated construction algorithms follow a greedy strategy by adding the solution components,

which means that at each step a solution component is chosen from a ranked list based on some heuristic information, instead of a simple random choice (Dorigo and Stützle, 2009). Constructive heuristics are usually problem-specific, non-iterative, and create one single solution by applying a set of rules based on problem-specific knowledge.

Search heuristics follow a certain search scheme that repeatedly examines many different solutions for a given problem in order to find better solutions (Zäpfel et al., 2010); cf. (Talbi, 2009).

Metaheuristics Search heuristics correspond to metaheuristics in a broader sense. Brucker (2007) defines any approach without a formal guarantee of performance as a heuristic. Informally, a metaheuristic states an algorithmic advancement of a simple heuristic, which is commonly defined as a rule of thumb that leads to near-optimal solutions without complete knowledge of the problem. There is no single and universal definition for the term metaheuristic, there exist numerous of them.

However, it seems that there evolved a widely accepted understanding of metaheuristics in academia: a metaheuristic is a general algorithmic framework that *a*) is generally problem-independent and applicable to a wide set of different problems, *b*) describes an iterative upper-level strategy that guides the operations of subordinate heuristics, *c*) combines different concepts for exploring and exploiting the search space (diversification and intensification), often facilitated by the use of randomness (Blum and Roli, 2003; Zäpfel et al., 2010).

Trajectory Methods vs. Population-Based Methods Most authors consistently distinguish between two classes of metaheuristics: trajectory methods (based on a single solution) and population-based methods (Blum and Roli, 2003); cf. (Luke, 2009; Talbi, 2009; Zäpfel et al., 2010; Marti and Reinelt, 2011; Baghel et al., 2012). The number of solutions used to obtain new solutions in every cycle of search is the distinguishing factor between the methods. More precisely, it is distinguished between search methods that operate on a single solution and those that operate on multiple solutions.

Trajectory methods obtain improved solutions by repeatedly modifying an existing solution during the search procedure, e.g. LS (Hill-Climbing), SA, TA, TS, GRASP, VNS, GLS, and ILS; cf. (Blum and Roli, 2003; Talbi, 2009; Baghel et al., 2012; Boussaïd et al., 2013).

Population-based methods, operating on a set of solutions, create improved solutions by recombining existing solutions e.g. EAs (GAs, EAs, EP, GP), SS, ACO, PSO and AIS; cf. (Blum and Roli, 2003; Talbi, 2009).

Hybrid Metaheuristics Beyond these basic groups, another class of algorithms has recently emerged: hybrid (meta)heuristics. Hybrid methods merge two or more search methods (i.e. metaheuristics) or only aspects of them as a new approach, in a sense that the resulting search procedure combines the strengths of different search methods (Baghel et al., 2012); cf. (Blum and Roli, 2003).

Hyper-Heuristics Hyper-heuristics cover automated methodologies for selecting or generating low-level (meta)heuristics. They intend to automate the design and adaptation of heuristic methods in order to produce more generally applicable search methodologies. The idea behind is that searching over a space of heuristics may be more effective than directly searching the underlying problem space. In this context, it is distinguished between two main categories: heuristic selection and heuristic generation (Burke et al., 2009). In the area of Evolutionary Computation (EC), this approach is referred to as meta-evolutionary approach.

Alternative Taxonomies Underpinning the observation that metaheuristics are subject to vital research, there exist different ways to classify metaheuristics, depending on the characteristics used for the distinction. Beyond subdividing between population-based methods and single-solution search, alternative taxonomies classify metaheuristics by distinguishing between *a*) nature-inspired and non-nature inspired methods, *b*) dynamic and static objective functions, *c*) one neighborhood structure and various neighborhood structures, *d*) memory usage and memory-less methods, *e*) deterministic and stochastic search, and *f*) iterative versus greedy search schemes (Blum and Roli,

2003; Talbi, 2009); cf. (Zäpfel et al., 2010; Marti and Reinelt, 2011) for more alternative taxonomies used to classify metaheuristics.

For example, Zäpfel et al. (2010) put emphasis on the strategy used to obtain new solutions and propose three groups of heuristic search algorithms: *a*) repeated solution construction, *b*) repeated solution modification, and *c*) repeated solution recombination. Here, repeated solution modification methods relate to trajectory methods and the methods based on repeated solution recombination refer to population-based methods. In contrast to the widely accepted two-group taxonomy (trajectory and population-based methods), Zäpfel et al. (2010) mention GRASP and ACO as parts of a third category, namely methods based on repeated solution construction. These methods always create new solutions from scratch (by construction). Unfortunately, important metaheuristics such as VNS, GLS, Evolution Strategies (ESs), and PSO remain unclassified in this taxonomy.

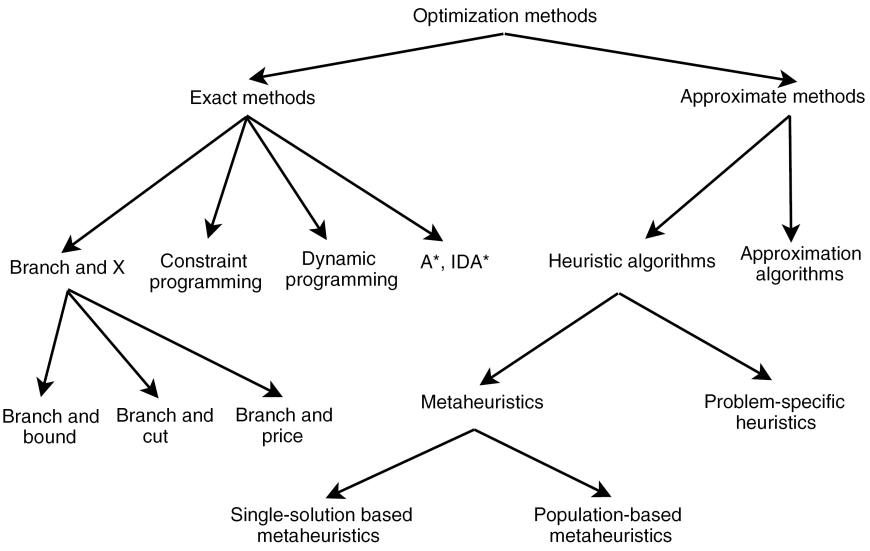


Figure 18: Taxonomy for methods solving combinatorial optimization problems (Talbi, 2009)

5.2 Complexity Theory

A key point in OR projects is to determine the difficulty of the underlying optimization problem, respectively of a particular COP that needs to be solved; cf. (Dorigo and Stützle, 2004). Complexity Theory has basically two main purposes. The first purpose is to determine the amount of computational resources required to solve a computational problem to optimality. The OR community discusses computational resources in the first place in terms of computational time and secondly in terms of space (memory), which also are referred to as time complexity and space complexity. The second purpose is to classify important problems according to their difficulty (Allender et al., 2010). Sipser (2006) gives a detailed introduction to the theory of computation, devoting much attention to Complexity Theory as a part of it.

Time Complexity The time complexity of a problem corresponds to the time complexity of the best (fastest) algorithm known to solve that problem. And an algorithm's time complexity is defined by its maximum (worst-case) number of computational steps required to optimally solve a problem instance with an arbitrary input size n (Talbi, 2009). Depending on the encoding scheme, e.g. binary or unary encoding, the input size n of a certain instance is defined by the length of the data string representing that particular instance. As a matter of fact, the unary encoding scheme leads to a larger instance size compared to that under binary encoding (Pinedo, 2008).

The $\mathcal{O}$-Notation The time complexity of an algorithm and thus of the relating problem, is given by the time complexity function, a mathematical function of the input size n. The purpose of the

time complexity function is not to obtain an exact computational step count, but to provide a maximum upper bound on the required step count in terms of an asymptotic (worst-case) analysis for arbitrary input sizes n. The $\mathcal{O}$-Notation is usually used to formalize the worst-case time complexity function of an algorithm or a problem.

With respect to OR problems, it is basically distinguished between two types of algorithms only: polynomial-time bounded and exponential-time bounded algorithms. A polynomial-time (bounded) algorithm is an algorithm with time complexity $\mathcal{O}(p(n))$, whereby $p(n)$ is a polynomial function of n. In other words, the time that the algorithm needs to optimally solve an instance of size n grows by means of the polynomial $p(n)$ in the worst-case, which means the algorithm is polynomially bounded. For example, $\mathcal{O}(n)$ grows linearly, $\mathcal{O}(n^2)$ quadratically and $\mathcal{O}(n^3)$ grows cubically; cf. (Reingold, 2010). Exponential-time (bounded) algorithms have a time complexity $\mathcal{O}(c^n)$, where c is a real constant strictly superior to 1. Here the computation time grows exponentially with the instance size n in the worst-case (Talbi, 2009); cf. (Dorigo and Stützle, 2004) and (Pinedo, 2008).

5.2.1 Decision Problems, Languages and Turing Machines

This section introduces some important terms in the context of OR and Complexity Theory, e.g. decision problems, languages and Turing machines, laying the basis for understanding the following sections.

Decision Problems In the context of Complexity Theory, COPs are stated in terms of decision problems with a yes/no answer. The arising decision problem asks to determine the existence of a solution for the corresponding COP with an objective value lower than a given threshold (for a minimization problem). Consequently, the time complexity of the algorithm that produces the correct yes/no answer to a decision problem then corresponds to the time complexity of the COP related to that decision problem. Clearly, the bigger the problem instance, the longer the algorithm needs to find the correct answer to the decision problem (Lenstra et al., 1977); cf. (Hedman, 2004; Pinedo, 2008).

Language Recognition Decision problems are strongly related to languages in the context of Theoretical Computer Science. A language recognition problem is a decision problem asking whether a given string (a word) belongs to a particular language. Thus, any COP can be stated in terms of a decision problem and decision problems can be interpreted as language recognition problems (Jiang et al., 2010). And the latter (language recognition problems) are solvable by deterministic and non-deterministic Turing machines (Lenstra et al., 1977).

Deterministic and Non-Deterministic Turing Machines The concept of Deterministic Turing Machine (DTM) defines the three basic steps in any mechanical computation: *a*) the ability to read and write on a storage medium, *b*) the ability to move on that medium, and *c*) the ability to make simple logical decisions. Those abilities, appropriately combined, result in an algorithm corresponding to a DTM, which is capable of deciding language recognition problems and thus solving decision problems. A DTM is basically a state machine, characterized by an algorithm that unambiguously defines a transition from one state to another (Jiang et al., 2010).

Beside DTMs, it is important to introduce the theoretical concept of Non-Deterministic Turing Machines (NTMs). NTMs do not model physical computation devices, they model a virtual machine without a physical representation in the real world. The reason is that a NTM is defined as a state machine with ambiguous state transitions, characterized by state transfer relations describing transitions from one state to a set of states, in contrast to the unambiguous transfer functions implemented in DTMs. However, NTMs represent an elementary component in Complexity Theory, since they are used to model real computational problems (Allender et al., 2010).

5.2.2 Complexity Classes $\mathcal{P}$ and $\mathcal{NP}$

In order to classify problems with respect to their complexity, there exist complexity classes that define polynomial and exponential bounds on time and space for deterministic and non-deterministic machines. The OR community only focuses on the two classes $\mathcal{P}$ (DTIME) and $\mathcal{NP}$ (NTIME),

which stand for deterministic polynomial time and non-deterministic polynomial time respectively (Allender et al., 2010). Both classes $\mathcal{P}$ and $\mathcal{NP}$ cover problems solvable in a number of steps bounded by a polynomial in the length of the input, but based on different types of Turing machines (DTM and NTM) (Lenstra et al., 1977).

The complexity class $\mathcal{P}$ is defined for decision problems that can be solved by a DTM in polynomial time, respectively in a number of steps bounded by a polynomial of problem instance size. In other words, a DTM or an algorithm needs polynomial time to produce the correct yes/no answer to a decision problem.

On the other hand, the complexity class $\mathcal{NP}$ is defined for decision problems that can be solved by a NTM in polynomial time. To phrase it differently, the class $\mathcal{NP}$ covers decision problems for which a DTM or an algorithm can verify a correct answer in polynomial time, independently of the way it was generated.

Summarizing, algorithms in $\mathcal{P}$ solve their corresponding problems in polynomial time, whereas algorithms in $\mathcal{NP}$ only verify an existing solution to the corresponding problem in polynomial time (Pinedo, 2008); cf. (Talbi, 2009; Dorigo and Stützle, 2004).

5.2.3 $\mathcal{NP}$-Completeness

The concept of $\mathcal{NP}$-completeness builds the formal basis to distinguish between easy and hard problems by defining a formal borderline between them.

An important concept underlying $\mathcal{NP}$-completeness is the concept of problem reduction. A problem P' reduces to problem P if for any instance of P' an equivalent instance of P can be constructed. We talk about polynomial reducibility if P is constructed in polynomial bounded time, which is then denoted by $P' \propto P$ (Lenstra et al., 1977); cf. (Pinedo, 2008).

According to Lenstra et al. (1977), a decision problem P is $\mathcal{NP}$-complete if $P \in \mathcal{NP}$ and $P' \propto P$ for every $P' \in \mathcal{NP}$. Informally, a decision problem P is $\mathcal{NP}$-complete if *a*) P is in the $\mathcal{NP}$ class and *b*) if all problems in $\mathcal{NP}$ polynomially reduce to P. A problem P, either a decision problem or an optimization problem, is called $\mathcal{NP}$-hard if it satisfies the second condition *b* only, i.e. all problems in $\mathcal{NP}$ polynomially reduce to P (Lenstra et al., 1977); cf. (Hedman, 2004; Pinedo, 2008; Talbi, 2009; Korte and Vygen, 2012). By this definition, any $\mathcal{NP}$-complete decision problem is $\mathcal{NP}$-hard. It has become common practice in the OR community to call optimization problems $\mathcal{NP}$-hard if their associated decision problems are $\mathcal{NP}$-complete (Brucker, 2007; Talbi, 2009).

At this point the question arises how it can be shown that every problem in $\mathcal{NP}$ reduces to a particular problem, especially considering the fact that there exists an infinite number of problems in the $\mathcal{NP}$-class. This credit must go to Cook (1971) who presents a generic reduction from Turing machines to the Satisfiability Problem (SAT), which in turn first establishes $\mathcal{NP}$-completeness for SAT (Leung, 2004). The master reduction from Cook (1971) constructs for any instance of $P \in \mathcal{NP}$ an equivalent boolean expression in conjunctive normal form in polynomial bounded time (Lenstra et al., 1977).

Starting from the fact that SAT is $\mathcal{NP}$-complete, Karp (1972) proofs $\mathcal{NP}$-completeness for a large number of COPs by polynomial reduction, e.g. directed Hamiltonian path, Partition and Knapsack (Leung, 2004). This strategy is used to locate the borderline that separates the easy problems (in $\mathcal{P}$) from the hard ($\mathcal{NP}$-complete) ones. Furthermore, it is said that a minor change in a problem parameter often transforms an easy problem into a hard one (Lenstra et al., 1977).

5.2.4 Strong vs. Ordinary $\mathcal{NP}$-Completeness

In order to determine the complexity of a problem, first a distinction between the membership of $\mathcal{P}$ versus $\mathcal{NP}$-completeness is made, respectively between easy and hard problems. But it is said that this is only a coarse indicator because there are significant differences in complexity within the class of $\mathcal{NP}$-complete problems. Those differences lead back to the encoding scheme used to specify the problem instance, i.e. unary versus binary encoding. Here it is distinguished between two classes of $\mathcal{NP}$-complete problems, respectively between unary $\mathcal{NP}$-complete problems ($\mathcal{NP}$-hard in the strong sense) and binary $\mathcal{NP}$-complete problems ($\mathcal{NP}$-hard in the ordinary sense).

A problem that is $\mathcal{NP}$-hard with respect to the binary encoding scheme but not under the unary encoding scheme is said to be $\mathcal{NP}$-hard in the ordinary sense or simply $\mathcal{NP}$-hard. The class of

$\mathcal{NP}$-hard problems in the ordinary sense can be solved in polynomial time under unary encoding, while it cannot be solved in polynomial time under binary encoding (pseudo-polynomial)

A problem that is $\mathcal{NP}$-hard with respect to the unary encoding scheme is said to be $\mathcal{NP}$-hard in the strong sense or strongly $\mathcal{NP}$. Either under unary or binary encoding for strongly $\mathcal{NP}$-hard problems there are no polynomial time algorithms known to this date (Leung, 2004; Pinedo, 2008).

Pseudo-Polynomial Referring to the Knapsack problem, this paragraph claries the term pseudopolynomial in the context of strong/ordinary $\mathcal{NP}$completeness. Knapsack is $\mathcal{NP}$-complete with respect to a binary encoding. But there exists a polynomial bounded algorithm based on DP with respect to unary encoding, respectively a pseudo-polynomial algorithm. Given the fact that unary $\mathcal{NP}$-completeness is not established, a binary $\mathcal{NP}$-complete problem allows an unary polynomial bounded solution (Lenstra et al., 1977). For the Knapsack problem, no algorithms are known for solving it with respect to a binary encoding scheme in time bounded by a polynomial in the input length. However, many practitioners consider Knapsack to be tractable. The reason is that algorithms are known which solve it in time bounded by a polynomial in the input length and the magnitude of the largest number in the given problem instance. Such algorithms whose time complexity depends on the input length and another factor related to the instance, are referred to as pseudo-polynomial algorithms (Garey and Johnson, 1978). In other words, an algorithm is said to be pseudo-polynomial if it is not polynomial with respect to binary encoding, but polynomial with respect to unary encoding (Lawler, 1977).

Summarizing, Knapsack was shown to the $\mathcal{NP}$-hard only under binary encoding, it was not shown to be $\mathcal{NP}$-hard under unary encoding. It is no contradiction that a problem is $\mathcal{NP}$-hard under the binary encoding scheme, but solvable in polynomial time under the unary encoding scheme, respectively in pseudo-polynomial time (Leung, 2004).

In contrast, the TSP is strongly $\mathcal{NP}$-hard and (unless $\mathcal{P} \neq \mathcal{NP}$) there is no exact pseudo-polynomial algorithm for any strongly $\mathcal{NP}$-hard problem (Korte and Vygen, 2012).

5.2.5 The $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$-Problem

As mentioned earlier, the class $\mathcal{P}$ is clearly a subclass of the class $\mathcal{NP}$. But one of the most important open issues in mathematics is the question whether $\mathcal{P} = \mathcal{NP}$ (Pinedo, 2008). The Clay Mathematics Institute[6] has chosen the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$-Problem as one of its seven Millennium Problems, each with a reward of one million dollars for their solution (Hedman, 2004). It is a matter of fact that either all $\mathcal{NP}$-complete problems are solvable in polynomial time or none of them. To this date, no single $\mathcal{NP}$-complete problem is shown to be solvable in polynomial time (Leung, 2004); cf. (Dorigo and Stützle, 2004; Pinedo, 2008). Lenstra et al. (1977) describe the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$-Problem by means of the master reduction from Cook (1971) and state that $\mathcal{P} = \mathcal{NP}$ if and only if SAT $\in \mathcal{NP}$, whereas SAT can be replaced by any $\mathcal{NP}$-complete problem. Consequently, if SAT $\in \mathcal{P}$ then every $\mathcal{NP}$-complete problem is in $\mathcal{P}$ and finally $\mathcal{P} = \mathcal{NP}$. However, in general the OR community considers the equality of $\mathcal{P}$ and $\mathcal{NP}$ as highly unlikely, which in turn means that a polynomial bounded algorithm for one and thus for all $\mathcal{NP}$-complete problems is highly unlikely to exist (Lenstra et al., 1977). See Figure 19 for a plausible visualization of the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$-Problem.

5.3 The Search Space

The performance of metaheuristics is closely linked to the structure of the underlying search space. Watson (2009) formally defines the search space $L = (S, N, F)$ by the combination of *a*) the state space S, *b*) the move operator N, and *c*) the objective function F. The search space can be seen as a vertex-weighted directed graph in which each vertex represents a state with a weight equal to the corresponding objective value and each edge describes a certain move operator that leads from one state to another.

[6] http://www.claymath.org/

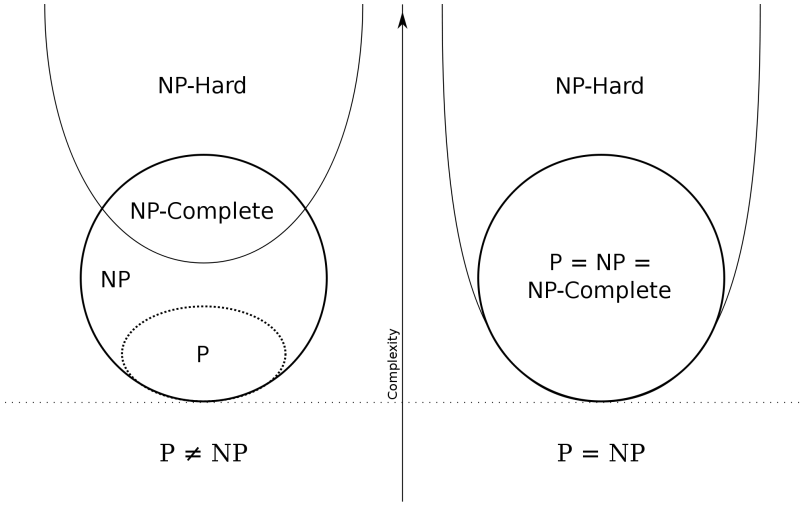


Figure 19: Euler diagram for $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-complete, and $\mathcal{NP}$-hard set of problems (wikipedia, 2014)

Terminology of Landscapes The OR community also refers to this graph as the fitness landscape (Watson, 2009). Commonly, descriptions of landscape structures in OR make use of geographical terms, e.g. valleys, plains, peaks, plateaus or basins. This geographical metaphor enables us to visualize the quality of solutions in two or three dimensions, where the altitude of a particular point in this system equals the objective value of the corresponding solution (Talbi, 2009). Watson (2009) defines structural characteristics of a fitness landscape in terms of: *a*) the number and/or distribution of local optima, *b*) the strength and size of local optima attractor basins, and *c*) the size and extension of the search space. Watson (2009) argues the lack of empirical evidence to evaluate the importance of the mentioned landscape features. Deb et al. (1997) design a number of fitness landscapes used as test functions in order to evaluate the performance of evolutionary algorithms. Based on sets of generated fitness landscapes, providing knowledge of optimal solutions and their neighborhood, they investigate the convergence properties of the tested algorithms. See Figure 20 for four examples of quality/objective functions. Watson (2009) and Talbi (2009) also present graphical representations of different landscape structures visualizing the search space.

Search Scheme vs. Search Space The interaction of a metaheuristic with the underlying fitness landscape defines its performance. Since metaheuristics define strategies for searching the landscapes, knowledge of their structure facilitates developing effective metaheuristics. This knowledge opens the opportunity to adapt a metaheuristic search scheme in a targeted manner (Watson, 2009). Incorporating landscape knowledge becomes in particular important with regard to the statement made in (Talbi, 2009), saying that not only different optimization problems correspond to different landscape structures, but also different instances of the same problem may be characterized by varying landscape structures.

5.4 Metaheuristic Design

Despite of the existence of a vast variety of different metaheuristics, all of them share some similarities. The basic design usually follows a black-box modeling approach, coupling the metaheuristic with the underlying problem and decoupling the metaheuristic search scheme from model-internal activities representing the behavior of the underlying problem. Another commonality (for modern metaheuristics) is a algorithmic design that balances intensification and diversification in the search behavior, enabling a metaheuristic to exploit (intensification) and explore (diversification) the search space in an efficient and effective way. Recent developments in metaheuristic research consider hybridization and parallelization as promising approaches, leading to improved search performance. At this point, this work refers to the work of Talbi (2009) who examines the design and implementation of metaheuristics more thoroughly.

A central issue for metaheuristics is the design of the appropriate neighborhood structure

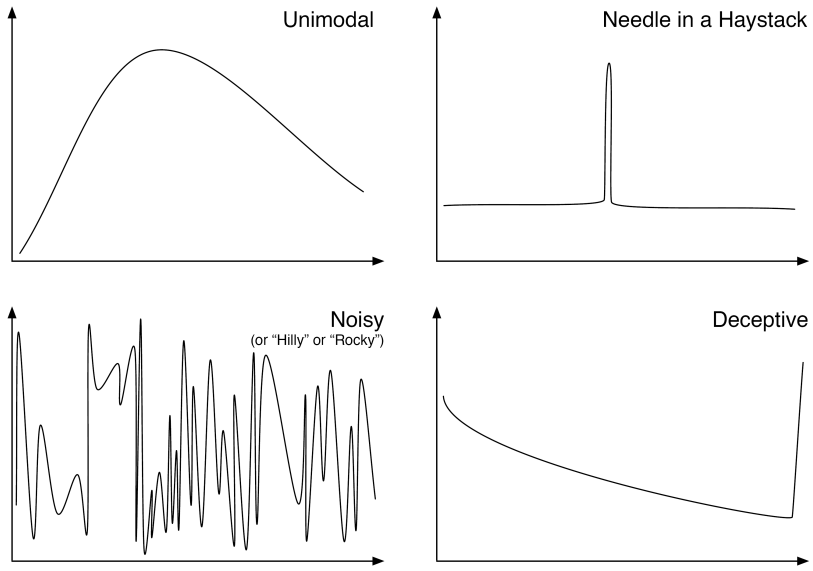


Figure 20: Four examples of quality/objective functions (Luke, 2009)

(Baghel et al., 2012). As stated earlier, as part of preparatory work for designing and implementing a metaheuristic for the problem at hand Hansen and Mladenović (2003) advise to analyze already proposed metaheuristics and to identify algorithmic key ingredients used for the search, in particular the structure of move operators (neighborhood design). Some other common issues include the use of memory, randomization, and dynamic parameter adjustment (Baghel et al., 2012). Battiti and Brunato (2009) refer to the strategy of dynamic parameter adjustment as to Reactive Search Optimization, describing the integration of machine learning techniques into search heuristics in terms of an online feedback loop for the self-tuning of critical parameters during the search process. Similarly, Bäck (1997b) proposes a self-adaptation approach for ESs, which dynamically evolves the strategy/control parameters during the search. A unified algorithmic view on metaheuristics is given in (Zäpfel et al., 2010).

Parejo et al. (2011) recently presented a comparative study of Metaheuristic Optimization Frameworks (MOFs). The metric used for evaluating the MOFs incorporates various important features that range from different metaheuristic techniques covered to documentation and user interface. There exist many MOFs that can speed up optimization projects and reduce their costs significantly; cf. (Talbi, 2009).

Properties One of the elementary properties characterizing metaheuristics is that they are generally problem-independent and applicable to a wide set of different problems (Zäpfel et al., 2010).

Blum and Roli (2003) outline some fundamental properties of metaheuristics: metaheuristics *a*) efficiently explore the search space in order to find (near-)optimal solutions, *b*) include techniques ranging from simple LS procedures to complex learning processes, *c*) are approximate and usually non-deterministic, *d*) may incorporate mechanisms to avoid getting trapped in confined areas of the search space (escape from local optima), *e*) permit an abstract level description in their basic concepts, *f*) use domain-specific knowledge in the form of heuristics controlled by the upper level strategy, and *g*) use search experience (memory) to guide the search in its more advanced variants. Hansen and Mladenović (2001) condense the desirable properties of metaheuristics into seven terms: *a*) simplicity, *b*) coherence, *c*) efficiency, *d*) effectiveness, *e*) robustness, *f*) user-friendliness, and *g*) innovation.

According to Prestwich (2008), many modern metaheuristics have the property of Probabilistic Approximate Completeness (PAC), which means that the probability of finding a solution tends to 1 as search time tends to infinity. PAC synonymously stands for metaheuristic's capability to

escape from any local minimum.

5.4.1 Black-Box Modeling

In contrast to mathematical programming techniques, e.g. LP and/or IP/MIP, no analytical formulation in terms of an unambiguous mathematical notation is required to apply metaheuristics. Common to all metaheuristics, the underlying model representation is simply considered as a black-box that returns an objective/quality value while considering the constraints given by the problem under study, based on the decision variables repeatedly modified by the metaheuristic applied. For the application of metaheuristics it does not matter how the objective value is produced (Talbi, 2009). See Figure 21 visualizing the described black-box concept.

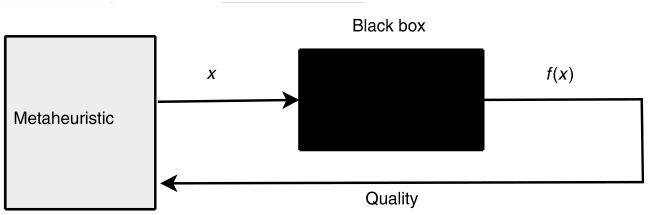


Figure 21: Black-box scenario for the objective function (Talbi, 2009)

Simulation(-Based) Optimization In those cases where simulation models are required to evaluate an objective function, i.e. due to stochastic effects, analytical methods based on explicit mathematical formulations are no longer an option (Talbi, 2009). Especially for stochastic problems, analytical models are often inadequate and simulation models provide a more suitable method (Hillier and Lieberman, 2001). Simulation-based optimization, or simply simulation optimization, describes the interaction between a simulation system, acting as a black-box that evaluates an objective function and an optimization method, e.g. a metaheuristic that manipulates the simulation system variables in a targeted manner. Beside analytical optimization models solved either by exact methods or approximate methods, simulation-based optimization powered by metaheuristics is another important branch in OR; cf. (Talbi, 2009). Fu et al. (2005) provide a descriptive review of the main approaches in the area of simulation optimization.

5.4.2 Intensification vs. Diversification

There are two basic concepts that determine the behavior of a metaheuristic: intensification and diversification. These two forces naturally act contrary to each other, but also complement each other at the same time (Blum and Roli, 2003). Intensification (exploitation) relates to metaheuristic components/activities that aim on (intensively) searching for new optima in a certain area of the search space. Diversification (exploration) means that a metaheuristic is capable of searching a maximum number of different regions of the search space. Exploring the search space by diversification avoids search procedures from concentrating on non-promising regions, which consist of solutions with low optimization potential. With respect to diversification, escaping from local optima is probably the most important capability for metaheuristics to provide (Talbi, 2009).

In literature there is a broad consensus about the observation that high-performing metaheuristics not only incorporate mechanisms supporting both the effects of intensification and diversification, but also provide an appropriate balance between them. In addition to this, a metaheuristic may adjust its parameters dynamically, emphasizing its preference for intensification or diversification depending on the progress of the search process (Blum and Roli, 2003); cf. (Zäpfel et al., 2010).

According to (Talbi, 2009), metaheuristics based on the single-solution approach are naturally more oriented on intensification, whereas population-based metaheuristics generally show a higher affinity to diversification.

Refer to Figure 22 showing the design space of a metaheuristic.

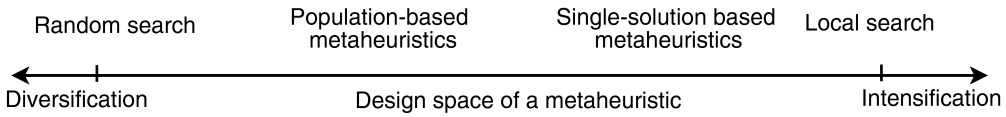


Figure 22: Design space of a metaheuristic (Talbi, 2009)

The I&D Frame Blum and Roli (2003) introduce the I&D frame in order to put algorithmic or functional components of metaheuristics (I&D components) into relation to each other, considering their effect on intensification and/or diversification during the search process. Despite of the fact that metaheuristics differ in their search strategies, the implemented mechanisms/components are all based on intensification and diversification, even if the paradigm behind is completely different. Blum and Roli (2003) emphasize that I&D components can have both an intensification and a diversification effect on the search procedure. The I&D frame describes a triangle where each corner corresponds to one of three basic characteristics associated with I&D components: objective guided (OG), non-objective guided (NOG), and randomness (R). The corner OG covers I&D components that are exclusively guided by the objective function. I&D components near to the corner NOG are guided by functions other than the objective function, e.g. mechanisms based on memory or problem-specific knowledge. The corner R refers to I&D components that simply employ randomness. It can be concluded that the corner OG stands for maximum intensification and minimum diversification, whereas the corners NOG and R come with maximum diversification and minimum intensification (Blum and Roli, 2003).

See Figure 23 that visualizes the I&D frame as described.

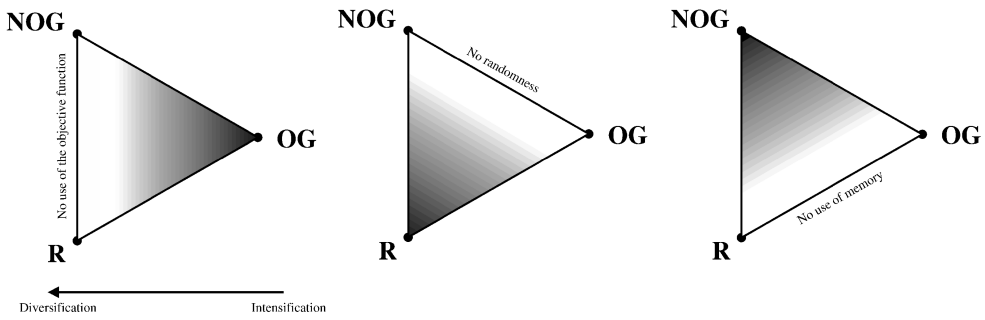


Figure 23: The I&D frame (Blum and Roli, 2003)

5.4.3 Hybridization

Hybrid metaheuristics refer to the idea of combining metaheuristics with other techniques for optimization. Hybridization aims to exploit the complementary character of different optimization strategies. In fact, combining an appropriate set of complementary algorithmic concepts can be the key for the design of high-performing search methods (Blum et al., 2011). The OR community shares the common understanding that well-designed hybrids often perform substantially better than classic metaheuristics. However, a more complex hybrid algorithm does not automatically perform better, since appropriate tuning of the methods parameter becomes more difficult as the method's complexity increases (Raidl et al., 2009).

In literature, basically two groups of hybrid metaheuristics are discussed: hybrids with other metaheuristics and hybrids with exact methods. Prominent examples for metaheuristics combined with exact methods are tree-search-based methods such as B&B, DP, LP, MIP, and CP (Blum et al., 2011); cf. (Raidl et al., 2009; Talbi, 2009; Baghel et al., 2012). Hybridizing metaheuristics with exact methods aims to combine the complementary strengths of complete and incomplete search (Prestwich, 2008). However, combining metaheuristics with each other is more popular, usually seeking to combine the advantages of trajectory methods and population-based methods into a single hybrid metaheuristic. By design population-based methods perform better in terms of

identifying promising search space areas (diversification), whereas trajectory methods are better in exploring promising search space areas (intensification) (Blum and Roli, 2003).

For more thorough insights into hybrid metaheuristics see (Blum et al., 2008) and (Talbi, 2009). Puchinger and Raidl (2005) present a survey on combining exact algorithms and metaheuristics to solve COPs. Blum et al. (2011) provide a recent survey on some of the most important topic lines of hybridization.

5.4.4 Parallelization

The idea of parallelizing metaheuristics is driven by two forces: the complexity of computational problems and the rapid development in technology of distributed computing systems. On one hand, despite of the fact that metaheuristics facilitate reducing computation times significantly, computational problems are often $\mathcal{NP}$-hard and remain computational expensive. Here, parallelism can help to reduce the computation time and increase the solution quality (Alba, 2005). Similarly, Talbi (2009) identifies four main goals of parallel and distributed computing: *a*) speed up the search, *b*) improve the quality of the obtained solutions, *c*) improve the robustness, and *d*) solve large-scale problems. On the other hand, the rapid development of technology in designing (multicore-)processors and networks leads to architectures suitable for the design and implementation of parallel metaheuristics (Talbi, 2009). Compared to single-solution based methods, population-based metaheuristics seem to be more appropriate for parallelism, since they already manage a set of parallel solutions by design (Luke, 2009).

For more detailed descriptions of distributed computing see (Kshemkalyani and Singhal, 2008). Crainic and Toulouse (2009) present a survey on parallel metaheuristic strategies, and Alba (2005) describe parallel approaches for *a*) LS, *b*) SA, *c*) TS, *d*) GRASP, *e*) VNS, *f*) models of EAs (in particular GAs, ESs, and GP), *g*) ACO, and *h*) SS.

5.5 Trajectory Methods

Trajectory methods, also referred to as single-solution based methods, are simply characterized by an iterative search scheme that operates on a single solution. Trajectory methods start from an initial solution and iteratively move from the current solution to another one in the search space, improving the initial solution step by step. They describe a trajectory in the search space. In contrast to population-based metaheuristics, trajectory methods are naturally more oriented on intensification than on diversification (Talbi, 2009). For more detailed descriptions see (Blum and Roli, 2003; Talbi, 2009; Baghel et al., 2012; Boussaïd et al., 2013), whereas Boussaïd et al. (2013) present a recent survey on single-solution based metaheuristics.

Trajectory methods mainly encompass: *a*) LS, *b*) SA, *c*) TA, *d*) TS, *e*) GRASP, *f*) VNS, *g*) GLS, and *h*) ILS (Blum and Roli, 2003; Talbi, 2009; Baghel et al., 2012; Boussaïd et al., 2013).

Local Search (LS) LS is likely the simplest metaheuristic method. Starting at a given initial solution, LS repeatedly replaces the current solution by a neighbor that improves the objective function. The search determinates when a local optimum is reached, meaning that all candidate neighbors are worse than the current solution with respect to the objective function (Talbi, 2009). See Figure 24 visualizing the concept of LS.

In this context, the term neighbor refers to a slightly modified solution obtained in each step of LS. And the finite set of modified solutions (neighbors) that can be obtained usually by many different modifications (moves) is also referred to as solution's neighborhood (Zäpfel et al., 2010). The definition of an appropriate neighborhood structure usually embeds problem-specific knowledge and is a crucial point for the performance of LS algorithms. The neighborhood of a solution is defined as the set of solutions that can be reached from that solution in one single step (Dorigo and Stützle, 2009).

LS is also referred to as iterative improvement, since each move is only performed if the resulting solution is better than the current solution (Blum and Roli, 2003). Another common term for LS is Hill-Climbing, where informally spoken, the algorithm climbs up the hill until the peak (local optimum) is reached (Luke, 2009).

LS methods have two main disadvantages: they get easily trapped in local minima and the result strongly depends on the initial solution (Dorigo and Stützle, 2009; Talbi, 2009). To overcome the problem of getting trapped in local minima, several techniques have been developed, adding mechanisms to LS (Blum and Roli, 2003). Another critical point is given by highly multi-modal objective functions, where LS is usually not an effective method to use. However, LS works well in the presence of more or less similar local optima with respect to the objective value and/or in those cases where not too many local optima exist (Talbi, 2009).

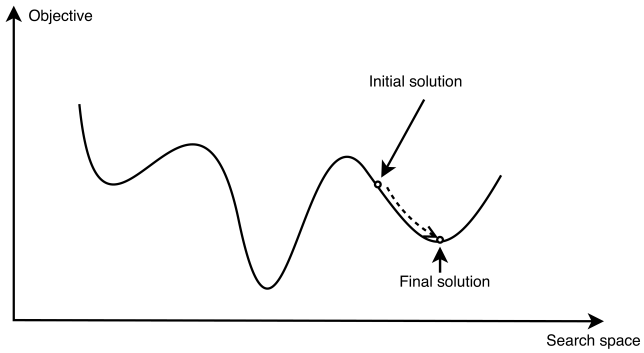


Figure 24: Basic Local Search (Talbi, 2009)

Best-Improvement vs. First-Improvement In general, it is distinguished between two basic types of LS, namely first-improvement and best-improvement. First-improvement LS explores the neighborhood and chooses the first solution improving the incumbent. Best-improvement first exhaustively explores the entire neighborhood and then returns the neighboring solution with the lowest objective function value (Blum and Roli, 2003).

First-improvement LS relies on the assumption that a certain neighborhood comprises many improving solutions and determining the best is not absolutely necessary. Advantageous of first-improvement LS is that it is less computationally expensive compared to best-improvement LS, but the solution quality may improve more slowly during the search and thus it may require more iterations to reach the nearest local optimum. The decision which scheme to prefer depends on the problem and the computational effort needed to generate one single neighbor, respectively the entire neighborhood (Zäpfel et al., 2010). Resende and Ribeiro (2009) observe that quite often both strategies lead to the same solution, whereas the first-improving strategy requires smaller computation times. Furthermore they notice that the best-improving strategy tends to converge to bad local optima in early phases of the search.

In (Hansen and Mladenović, 2003; Dorigo and Stützle, 2009; Hansen et al., 2009; Talbi, 2009; Zäpfel et al., 2010) one will find pseudocodes for the LS method and its variants with respect to the best-improvement and first-improvement strategy. For parallel designs and implementations of LS see (Alba, 2005).

Section 8.3.3 deals with experiments investigating the two LS strategies embedded in different VNS variants.

Escaping from Local Optima The disadvantage of LS is its convergence toward local optima. See Figure 25 for an adequate visualization of local and global optima. The strategies that have been proposed to escape from local optima are classified in four groups: *a*) iterating from different initial solutions, *b*) accepting non-improving neighbors, *c*) changing the neighborhood, and *d*) changing the objective function (Talbi, 2009).

The strategies that iterate from different initial solutions are commonly encompassed by the group of multi-start methods, e.g. ILS and GRASP. Approaches based on the idea of accepting non-improving neighbors allow degrading moves from one solution to another in order to escape from a local optimum, e.g. SA, TS and TA. Another idea is to change the neighborhood structure during the search, e.g. VNS strategies. The fourth group frames strategies perturbing the objective function

in order to enable the search to leave valleys around local optima, e.g. GLS. Beyond changing the neighborhood structure and changing the objective function, relaxing certain constraints given by the underlying optimization problem at hand states another strategy that modifies the search space (Talbi, 2009).

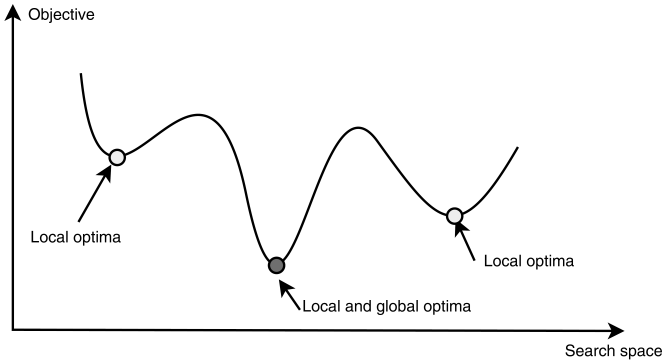


Figure 25: Local and global optima (Talbi, 2009)

Multi-Start Methods Especially with the example of multi-start methods, the meaning of the desired balance between diversification and intensification becomes obvious. Since LS is naturally focused on intensification by extensively exploiting the neighborhood of solutions, there is a striking need for diversification to overcome local optimality. Multi-start methods achieve diversification by repeatedly restarting LS from a new initial solution once a region has been entirely explored. Hence, the solution space is strategically sampled where each (LS) iteration leads to a local optimum and the best among them is considered as the best overall solution of the algorithm. The basic multi-start strategy in its simplest version restarts LS with randomly generated initial solutions (Marti et al., 2009; Marti and Reinelt, 2011; Marti et al., 2013).

Termination Conditions Beyond simple LS performing a deterministic search that finishes with a local optimum, most of the metaheuristics perform a non-deterministic search that incorporates randomness. They need for appropriate stopping conditions preventing endless search. Possible termination conditions include: *a*) the maximum computing time, *b*) the maximum number of iterations, *c*) a solution s with $f(s)$ less than a predefined threshold value, or *d*) the maximum number of iterations without any improvements (Blum and Roli, 2003).

5.5.1 Iterated Local Search (ILS)

ILS is essentially a multi-start procedure that focuses the search on already known local optima. Its success lies in the biased sampling of this set of local optima. ILS performs better than random restart, even in its most naive implementation (Lourenco et al., 2009). ILS can be seen as an improved version of basic LS with random restarts. The basic idea underlying ILS is that local optima often exist in near proximity to each other. The assumption/observation is that restarting LS in the near of already known local optima often outperforms just trying new locations entirely at random (Luke, 2009).

The basic idea of ILS stems from the observation that the performance of a LS method strongly depends on the initial solution. The basic multi-start LS randomly generates initial solutions in every restart cycle and thus successive initial solutions are neither related to each other nor related to the local optima found during the search. ILS improves the basic multi-start LS by considering a perturbed local optimum as initial solution in the succeeding restart cycle. After performing LS to an initial solution, the resulting local optimum is stochastically perturbed, followed by another LS cycle using the perturbed local optimum as initial solution (Talbi, 2009). Figure 26 shows the basic principle of ILS; cf. (Blum and Roli, 2003; Lourenco et al., 2009) for similar visualizations. Talbi (2009) provides pseudocode for ILS.

The degree of perturbation is a crucial point for the performance to expect and indeed interacts with the structure of the search space. A too low degree of perturbation might not enable the system to escape from local optima, while a too high degree results in a search behavior similar to random restart LS (Blum and Roli, 2003).

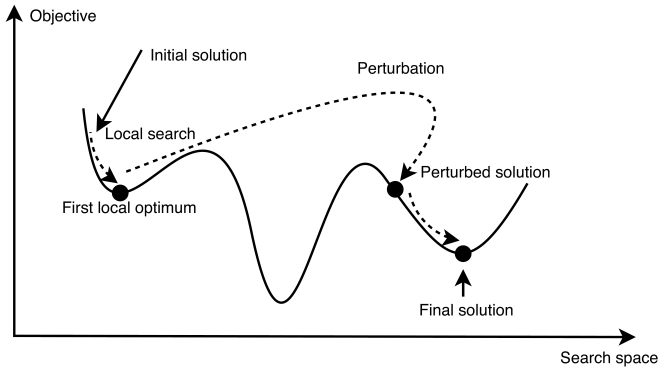


Figure 26: Iterated Local Search (ILS) (Talbi, 2009)

5.5.2 Guided Local Search (GLS)

In contrast to other metaheuristics that simply use randomness to escape from local optima, GLS exploits structural information of solutions obtained during search. GLS selects features present in a local optimum and augments the objective function with penalties corresponding to the selected features.

At the beginning all penalties are initialized to zero and whenever the search settles in a local optimum, GLS increases the penalties for those features present in the current optimum. As a result of the modified objective value worsened by feature penalties, the search leaves the current optimum and moves to another, probably better one. The penalties are updated dynamically during the search. Refer to Figure 27 for a graphical representation of the GLS concept; cf. (Blum and Roli, 2003) for another picture.

In contrast to the majority of metaheuristics, the diversification process in GLS is directed and deterministic rather than undirected and random (Voudouris et al., 2009); cf. (Blum and Roli, 2003; Talbi, 2009; Zäpfel et al., 2010). The critical point is to define the solution features that capture important structural properties of the solutions.

Talbi (2009) gives pseudocode for GLS.

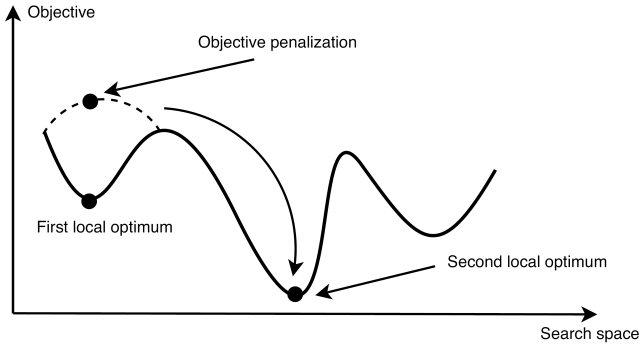


Figure 27: Guided Local Search (GLS) (Talbi, 2009)

5.5.3 Simulated Annealing (SA)

SA basically performs in the same way as basic LS, but allows non-improving moves according to a probability function (Marti and Reinelt, 2011). Improving solutions are always accepted, while a fraction of non-improving solutions are accepted in order to escape from local optima. The probability of accepting non-improving solutions depends on a temperature parameter decreasing with each iteration (Nikolaev and Jacobson, 2009). The probability of accepting non-improving moves depends on two factors: the degree of deterioration in terms of the objective value difference and the temperature. The probability function combines the degree of deterioration and the temperature: the higher the deterioration in the objective value, the lower the probability to accept a non-improving move at a fixed temperature. Hereby, the probability of accepting worsened solutions decreases with decreasing temperature controlled by the cooling schedule. In the beginning of the search, the probability of accepting non-improving moves is high and thus permits the exploration of the search space, but slowly decreases with ongoing search and finally leads the search to converge to a (local) minimum (Blum and Roli, 2003). In the end, when the temperature is adequately low, SA only allows improving moves and thus stops at a local optimum (Baghel et al., 2012). See Figure 28 for a graphical example.

The SA method has two important control components: the acceptance probability function deciding whether to accept a non-improving solution at a certain iteration or not and the cooling schedule determining the temperature at each step (Talbi, 2009). The cooling schedule is a critical issue to parametrize. If the cooling happens too rapid, i.e. the temperature almost immediately approaches zero, non-improving moves become very unlikely and SA degrades to basic LS. On the contrary, a too slow cooling schedule means that SA would undesirable perform similar to random search in the beginning of the search (Zäpfel et al., 2010).

Alba (2005) describe parallel SA concepts. For pseudocodes see (Talbi, 2009; Zäpfel et al., 2010; Zomaya and Kazman, 2010).

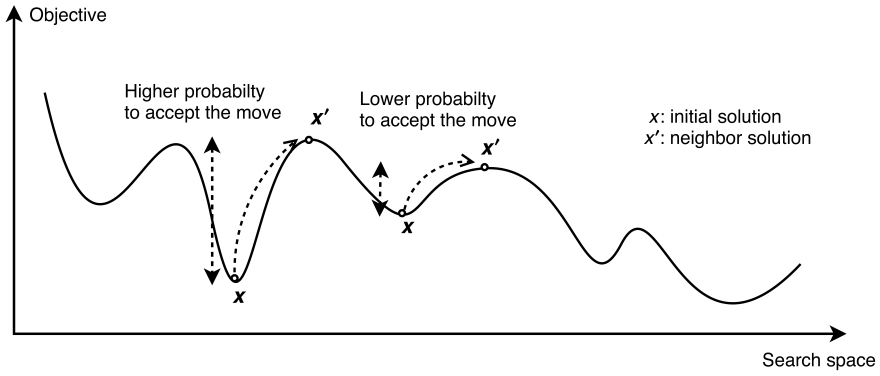


Figure 28: Simulated Annealing (SA) (Talbi, 2009)

5.5.4 Threshold Accepting (TA)

TA may be viewed as the deterministic variant of SA. TA allows non-improving moves in a sense that an accepted solution in each step is not worse than the current solution by more than a given threshold. The threshold parameter, corresponding to the temperature in SA, is updated following an annealing schedule decreasing the threshold with the number of iterations performed during the search (Talbi, 2009).

The threshold value is used to decide whether worse solutions are accepted or not. By accepting deteriorations in solution quality, an LS algorithm becomes capable of leaving local optima in order to find new (better) ones (Zäpfel et al., 2010).

It is said that TA performs faster than SA, because TA is not burdened by the generation of random numbers and exponential functions, which are used to decide for or against accepting a worsened solution within the SA concept (Talbi, 2009).

Talbi (2009) and Zäpfel et al. (2010) present pseudocodes for TA.

5.5.5 Tabu Search (TS)

TS basically employs best-improvement LS enhanced with a memory structure, accepting non-improving solutions in order to escape from local optima. The characteristic feature of TS is the use of memory.

TS performs almost identical to basic LS, moving from one improving solution to the next until a local optimum is reached. When the deterministic LS ends up with a local optimum, TS allows a non-improving move by selecting the best solution out of all non-improving solutions in the local optimums neighborhood previously investigated by best-improvement LS. It cannot be excluded that this strategy generates cycles in a sense that previous visited solutions are selected again. Therefore, TS memorizes the recent search trajectory and discards the neighbors that have been previously visited, forcing the search to explore the search space beyond the area in vicinity of the local optimum visited as last (Talbi, 2009). TS updates the memory (tabu list) with recently considered solutions and refuses the search to return until they are sufficiently far in the past (Luke, 2009). The search terminates if a stopping criterion is met or if all the solutions in the current neighborhood are forbidden by the tabu list (Blum and Roli, 2003).

Depending on the tabu list length, TS can be time and space consuming, since storing all visited solutions needs space and checking the tabu list in every cycle requires time (Talbi, 2009). The length of the tabu list is the important control parameter. Small tabu lists let the search concentrate on small areas of the search space, whereas large tabu lists force the search to explore larger regions of the search space (Blum and Roli, 2003).

To generalize memory-based methods, the term Adaptive Memory Programming (AMP) describes methods that use advanced memory strategies to guide a search. In this context, TS is covered by AMP (Marti et al., 2013).

In (Talbi, 2009; Zäpfel et al., 2010) one will find pseudocodes for the TS algorithm. Alba (2005) describe approaches for its parallelization.

5.5.6 Variable Neighborhood Search (VNS)

VNS was first mentioned in (Mladenović and Hansen, 1997) and then examined in different variants in (Hansen and Mladenović, 2001, 2003; Hansen et al., 2001, 2009). The basic idea behind is a systematic change of the neighborhood during the search, typically established in two alternating search phases, a descent LS phase and a randomized perturbation phase (shaking). This introductory definition holds for the basic VNS scheme and the general VNS scheme discussed later. In compliance with the concept of intensification and diversification, VNS basically exploits deterministic LS in a descent phase investigating local optima, whereas the perturbation phase employs randomness in order to escape from the corresponding valley (Hansen et al., 2009).

According to (Hansen et al., 2009), VNS is based on three simple facts: *a*) a local minimum with respect to one particular neighborhood structure is not necessarily so for another, *b*) a global minimum is a local minimum with respect to all possible neighborhood structures, and *c*) for many problems, local minima with respect to one or several neighborhoods are relatively close to each other.

Section 8.3 deals with experiments investigating different VNS variants with varying settings in order to determine favorable method settings.

Variants The concept of VNS is reflected in four basic variants: *a*) Variable Neighborhood Descent (VND), *b*) Reduced Variable Neighborhood Search (RVNS), *c*) Basic Variable Neighborhood Search (BVNS), and *d*) General Variable Neighborhood Search (GVNS) (Hansen and Mladenović, 2003; Hansen et al., 2009).

In the concept of VNS, VND and RVNS are considered as the basic building blocks, whereas BVNS and GVNS describe more sophisticated two-level compositions of them. VND is designed as a deterministic LS scheme aimed on intensification. It is used in the descent search phase of GVNS. In contrast, RVNS is entirely stochastic aiming for diversification and establishing the randomized perturbation phase (shaking) in GVNS and BVNS. Despite of their simplicity, both VND and RVNS justify to be applied as independent search methods.

Beyond the four basic schemes (VND, RVNS, BVNS and GVNS), especially for solving large problem instances, extensions have been proposed, in particular Skewed Variable Neighborhood

Search (SVNS) and Variable Neighborhood Decomposition Search (VNDS). VNDS describes a two-level search scheme enhanced through decomposition, aimed at increasing the precision and at reducing the solution time for decomposable problems. SVNS is designed to efficiently explore valleys far from the incumbent solution, employing a concept for distances between solutions (Hansen and Mladenović, 2001).

In this context, Pisinger and Ropke (2009) propose Large Neighborhood Search (LNS), which belongs to the class of Very Large Scale Neighborhood Search (VLSN) algorithms. VLSN algorithms are based on the observation that searching a large neighborhood results in finding local optima of high quality compared to the search in smaller neighborhoods. Indeed, searching a large neighborhood is time consuming and hence various filtering techniques are used to limit the search. In LNS the neighborhoods are implicitly defined by methods (often heuristics), used to gradually improve an initial solution by alternately destroying and repairing the incumbent solution.

For a parallel design of VNS see (Alba, 2005).

Refer to Figure 29 visualizing the basic concept of VNS.

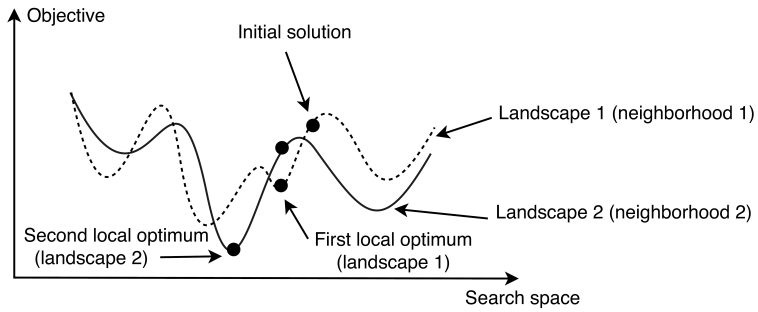


Figure 29: Variable neighborhood Search (Talbi, 2009)

5.5.7 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is a multi-start metaheuristic consisting of two phases: construction and LS. Beginning with a randomized construction phase building a feasible solution, LS investigates its neighborhood in the second phase until a local minimum is found. The best solution among all multi-start iterations is kept as the final result (Feo and Resende, 1995; Resende and Ribeiro, 2003).

Talbi (2009) notices that the search iterations are completely independent from each other and hence there is no search memory as in TS; cf. (Baghel et al., 2012). One will find annotated bibliographies of the GRASP literature in (Festa and Resende, 2002, 2009b,a).

Typical for construction algorithms, GRASP starts from an empty solution. The construction phase builds an initial solution by iteratively adding partial solution components, selected from a Restricted Candidate List (RCL) in each iteration. The construction phase is characterized by both a greedy and a probabilistic aspect. The greedy aspect manifests itself in the way the RCL is created/updated in every construction step. The probabilistic aspect corresponds to the selection strategy, randomly choosing a partial solution element from the RCL in each step. In each step of the construction phase, the RCL is formed by the set of the remaining solution components ranked according to a greedy evaluation function, which is usually a simple heuristic incorporating problem-specific knowledge. The solution element that is finally used to extend the existing partial solution is then chosen randomly from the RCL. The construction phase plays an essential role in creating biased starting solutions of high-quality for the LS (Resende and Ribeiro, 2009).

The length of the RCL is an important control parameter, determining the degree of variation established in the construction phase. In the case that the RCL length equals one, the construction phase would repeatedly return an identical solution entirely determined by the evaluation function used to update the RCL. Considerably too long RCL length would degrade GRASP to a simple random-restart LS procedure (Zäpfel et al., 2010).

The neighborhood search in the LS phase may be implemented using either a best-improving or a first-improving strategy. It was observed that quite often both strategies lead to the same

final solution, whereas the first-improving strategy requires smaller computation times. In addition, that best-improving strategy tends to converge to bad local optima in early phases of the search (Resende and Ribeiro, 2009).

For GRASP pseudocodes see (Feo and Resende, 1995; Resende and Ribeiro, 2009; Talbi, 2009; Zäpfel et al., 2010). Alba (2005) discuss parallel GRASP.

5.6 Population-Based Methods

In contrast to trajectory methods based on a single-solution, population-based methods iteratively operate on a set of numerous solutions, improving the solution set usually referred to as population. As they deal with a population of solutions in each iteration, they provide a natural, intrinsic way for the exploration of the search space (Blum and Roli, 2003). Similarly, Talbi (2009) also emphasizes that that population-based metaheuristics in comparison to trajectory methods generally show a higher affinity to diversication. There is a characterizing feature of population-based methods that makes them differ from parallelized trajectory methods: the solutions within a population usually affect each other while the search progresses (Luke, 2009).

EC covers a popular set of population-based metaheuristics, jointly characterized by the underlying idea to adapt principles from genetics and evolution in biology. An algorithm belonging to EC is referred to as EA. The GA is probably the most popular among them (Luke, 2009). Beyond EC, the class of population-based metaheuristics also covers *a*) ACO, *b*) PSO, *c*) SS, *d*) Bee Colony Optimization (BCO), *e*) Estimation of Distribution Algorithms (EDAs), and *f*) AISs (Blum and Roli, 2003); cf. (Talbi, 2009; Baghel et al., 2012). As the method's names indicate, nature-inspired approaches play a predominant role in the class of population-based metaheuristics.

The following sections are devoted to EAs, ACO, PSO, and SS. Since BCO, EDAs, and AISs are not covered in the next sections, the reader is referred to (Teodorovic, 2009) for BCO, to (Blum and Roli, 2003) for EDAs, and to (Greensmith et al., 2009) for AISs.

Swarm Intelligence (SI) In this context, Swarm Intelligence (SI) is often discussed in connection with a considerable amount of population-based methods, emphasizing the idea of producing computational intelligence by adapting concepts of social interaction in swarms that exist in nature (Boussaïd et al., 2013). A swarm is considered as a group of cooperating individuals/agents with a certain behavioral pattern to achieve some goal(s). The most discussed SI models include ACO and PSO. SI models exhibit three general properties: *a*) each entity of the swarm is made of a simple agent, *b*) communication among agents is generally indirect and short, *c*) cooperation among agents is realized in a distributed manner without a centralized control mechanism (Lim et al., 2009). For more thorough descriptions and recent innovations in SI see (Bonabeau et al., 1999; Lim and Jain, 2009).

5.6.1 Evolutionary Computation (EC)

EC employs the theory of evolution in nature as an algorithm. EC basically covers *a*) GAs, *b*) EP, *c*) ESs, and *d*) GP (Ashlock, 2006); cf. (de Jong et al., 1997; Bäck, 1997a; Blum and Roli, 2003; Talbi, 2009; Marti and Reinelt, 2011).

An algorithm belonging to EC is referred to as EA. In EC the set of solutions is referred to as population. A single solution is called individual. Recombined individuals are known as parents and the resulting individual is called offspring or child (Zäpfel et al., 2010).

EC is inspired by the concept of evolution, nature's capability to evolve living beings well adapted to their environment. At each iteration/generation, EAs apply a number of operators to the individuals of the current population in order to generate the individuals of the next generation's population (reproduction). There exist two types of operators used to generate new individuals: recombination and mutation operators. The driving force in EC is given by appropriate selection and replacement strategies, preferring individuals with a higher fitness (solutions with better objective values). The fitter the individual, the higher is the probability to be chosen as a member of the next generation's population. This notion of competition corresponds to the principle of survival of the fittest in natural evolution (Blum and Roli, 2003). The basic procedure that produces a new population in each generation is visualized in Figure 30.

The main search components for designing an evolutionary algorithm are as follows: *a*) representation, *b*) population initialization, *c*) objective function, *d*) selection strategy, *e*) reproduction strategy, *f*) replacement strategy, and *g*) stopping criteria (Talbi, 2009).

For more thorough insights into EC see (Bäck, 1997a; Ashlock, 2006; Lachowicz and Miekisz, 2009), whereas Ashlock (2006) gives a pseudocode for EAs among others. For recent surveys and thorough descriptions of EC with focus on multi-objective optimization see (Coello Coello, 1998a,b, 1999; Zitzler et al., 2001; Ghosh and Dehuri, 2004; Obayashi, 2007; Zhou et al., 2011). Alba (2005) describe parallel models of EAs.

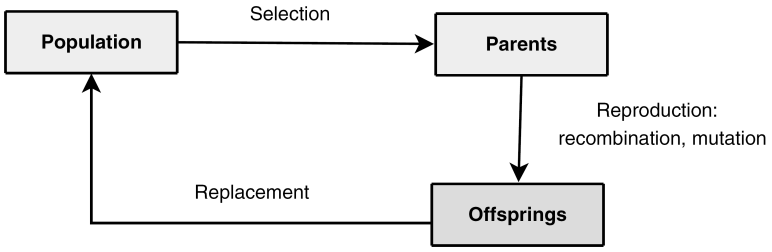


Figure 30: A generation in Evolutionary Algorithms (Talbi, 2009)

Evolutionary Programming (EP) EP was originally proposed to operate on discrete representations of finite state machines, but most of the present variants are used for continuous optimization problems (Blum and Roli, 2003). The common idea behind EP, as in ES, is to use mutation and selection, without employing recombination (Reeves, 2009); cf. (Bäck, 1997a; Porto, 1997). This emphasis on mutation operations generates diversity among the population of solutions, prevents entrapment in local minima, and maintains a high degree of correlation between parent and offspring behavior (Porto, 1997). In contrast to GAs, waiving recombination operators in the concept of EP basically stems from the realization that a sum of optimal parts rarely leads to an optimal overall solution. EP (and ESs) allow(s) for simultaneous modification of all decision variables by mutation at the same time (Porto, 1997).

Evolutionary Strategies (ESs) As EP, most variants of ESs are used for continuous optimization problems (Blum and Roli, 2003). In contrast to GAs, the traditional ES does not make use of recombination, instead they only use mutation for reproduction (Zäpfel et al., 2010); cf. (Reeves, 2009). Underpinned by the facts that the ES exclusively relies on mutation and initially even did not use a population of solutions, the idea behind ESs is generally closer in concept to neighborhood search methods, respectively variants of LS (Reeves, 2009). Another specificity is given by the selection operator that is originally deterministic. In addition, a distinguishing characteristic is that ESs also evolve control parameters in a kind of self-adaptation during the search process (Bäck, 1997a); cf. (Talbi, 2009). Since they are originally designed for continuous optimization problems, the self-adapting behavior automatically adjusts the step widths that determine the amount of variation of the parameters during the search (Zäpfel et al., 2010). Rudolph (1997) gives a pseudocode for ESs and Alba (2005) describe parallel variants.

Genetic Algorithms (GA) GAs are originally proposed and mainly applied to solve COPs (Blum and Roli, 2003). It is fair to say that GAs are probably the most discussed methods in the area of EC and encapsulate the most important concepts present in EC (Reeves, 2009). The concept of GAs is characterized by a strong emphasis on recombination as the most important search operator, whereas mutation is used with small probabilities in order to modify the individuals only slightly. The selection operator is usually stochastic. GAs often rely on a binary representation of individuals (Bäck, 1997a). This emphasis on recombination draws contrast to the concepts of EP and ES (Reeves, 2009); cf. (Eshelman, 1997). GAs usually use a probabilistic selection operator that is originally the proportional selection (Talbi, 2009). In the concept of GAs an individual representing a solution is usually encoded as a string, which is originally of binary form

(Reeves, 2009). It turned out that the binary encoding is impractical for many problems and the solution representation has been extended to include character based encoding, real-valued encoding, and even tree representations (Marti and Reinelt, 2011); cf. (Eshelman, 1997). In addition to common EA parameters (population size, initialization methods, fitness definition, selection and replacement strategies, and recombination and mutation operators), recent approaches employ additional information, e.g. the aging of individuals (Reeves, 2009). For GA pseudocode examples see (Eshelman, 1997; Reeves, 2009; Zäpfel et al., 2010). Alba (2005) investigates parallel GAs.

Genetic Programming (GP) The central idea of GP is to employ concepts of EC for automatically evolving executable computer programs (Langdon et al., 2009). In contrast to other EAs, in GP the evolving individuals are themselves programs that solve a given task. The theory of GP is considerably less developed than in ES and GA (Talbi, 2009). An individual in GP that represents the program to evolve is usually expressed as syntax tree rather than as lines of code. The variables and constants in the program to evolve are leaves of the tree, called terminals, whereas arithmetic operations are internal nodes of the tree, referred to as functions (Langdon et al., 2009). The major difference to other EAs is that the population and the individuals are stored in variable-sized trees (Ashlock, 2006). In GP evaluating the fitness of an individual means to execute the evolved computer programs in order to determine their quality with respect to a given task (Kinnear et al., 1997). Langdon et al. (2009) give a GP pseudocode and Alba (2005) deal with parallel GP.

Meta-Evolutionary Approach For the design of a particular EA various control parameters need to be defined, i.e. population size, initialization methods, fitness definition, selection and replacement strategies, and recombination and mutation (Reeves, 2009); cf. (Talbi, 2009).

Freisleben (1997) mentions a meta-evolutionary approach, describing a two-level concept in which a meta-level EA operates on a population of base-level EAs solving the problem at hand. The idea behind is to use a meta-level EA that determines the best settings for the base-level EAs. More in general and not exclusively linked to EAs, this approach is also combined with the term hyper-heuristics.

5.6.2 Ant Colony Optimization (ACO)

ACO is inspired from the foraging behavior of real ants, which indirectly communicate with each other via pheromone trails (Dorigo and Blum, 2005). The ACO concepts stems from the observation that an ant colony is able to discover the shortest path to locations of food sources in vicinity of their nest, even in complex scenarios with multiple paths. Despite of the fact that an individual ant has very limited skills, the whole ant colony successfully copes with the task of finding the shortest paths to food sources, enabled by a cooperation that is characteristic to SI (Zäpfel et al., 2010).

The ACO is considered as a distributed, stochastic search method based on the indirect communication of a colony of (artificial) ants, mediated by (artificial) pheromone trails. ACO is a population-based metaheuristic with single individuals (ants) that use a probabilistic construction scheme to create solutions, while exploiting an indirect form of memory of previous performance. In particular, the iteratively modified pheromone trails reflecting the ants search experience are used to probabilistically construct solutions (Dorigo and Stützle, 2004).

The randomized construction heuristic implemented in ACO makes probabilistic decisions as a function of some heuristic information about the problem instance and the dynamically acquired search experience (memory) in terms of artificial pheromone trails. In contrast to other construction methods (GRASP), ACO employs a cumulated search experience for constructing solutions. Facilitating diversification, a stochastic component in ACO enables the ants to build a wide variety of different solutions compared greedy construction heuristics (Dorigo and Stützle, 2009).

The use of ACO is particularly proposed to solve $\mathcal{NP}$-hard problems, dynamic shortest-path problems, and problems in which the computational architecture is spatially distributed (Dorigo and Stützle, 2004). Dorigo and Stützle (2009) list numerous ACO applications.

Dorigo and Stützle (2009) point out that coupling ACO with LS algorithms improves ACO's performance as the two approaches perform complementary. On one hand, ACO serves as a generator for appropriate initial solutions for LS algorithms. On the other hand, the improved

results by LS are used to enhance the search experience (pheromone trails), which in turn facilitates the construction of new initial solutions. Neumann et al. (2009) also discuss ACO's hybridization with LS.

For ACO-pseudocode see (Dorigo and Blum, 2005; Dorigo and Stützle, 2009; Talbi, 2009; Zäpfel et al., 2010). Alba (2005) investigate parallel ACO concepts.

5.6.3 Particle Swarm Optimization (PSO)

PSO is another stochastic population-based metaheuristic inspired from SI, characterized by a coordinated behavior using local movements emerges without any central control (Talbi, 2009). PSO is a stochastic optimization technique similar to EC, but unlike EAs, PSO does not utilize recombination, mutation or selection for producing new solutions. Instead, PSO establishes a form of directed modification in response to new discoveries about the search space, while maintaining a single static set of solutions (Luke, 2009); cf. (Lim et al., 2009).

Initially, a set of solutions is randomly distributed in the search space, also referred to as particle swarm. Every particle knows *a*) its actual value determined by an objective function, *b*) its own best objective value from history (locally best solution), *c*) the best objective value of the whole swarm (globally best solution) and *d*) its own velocity (Zäpfel et al., 2010); cf. (Baghel et al., 2012). In each iteration cycle of the search, every particle is modified under consideration of its actual objective value, its locally best solution, the globally best solution, its own velocity, and a bit of random noise (Luke, 2009). Consequently, the entire swarm moves in the direction of the globally best value, and thus explores the solution space in a directed fashion (Zäpfel et al., 2010). Figure 31 visualizes the described concept.

PSO is explicitly proposed for multi-modal problems, whereas the search space has multiple global optima or one global optimum with many local optima (Barrera and Coello Coello, 2009). But originally, PSO was designed to solve continuous optimization problems (Talbi, 2009). Reyes-Sierra and Coello Coello (2006) present a survey on PSO solving problems with multiple objectives. Talbi (2009) presents a pseudocode.

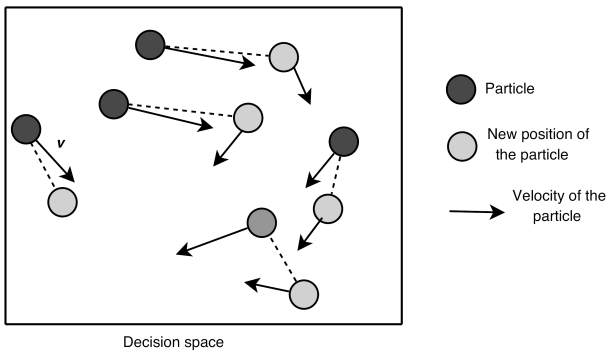


Figure 31: Particle Swarm Optimization (PSO) (Talbi, 2009)

5.6.4 Scatter Search (SS)

SS is a population-based metaheuristic operating on a set of solutions setting emphasis on recombination. SS was first introduced as a heuristic for IP problems (Marti and Reinelt, 2011). According to related EC/OR literature, SS is not covered by EC, but nevertheless considered as an evolutionary metaheuristic that employs deterministic recombination in a population of solutions in order to obtain better solutions (Resende et al., 2009). The specificity of SS is to establish a balance between diversification and intensification by explicitly controlling the diversity of the population (Zäpfel et al., 2010).

In contrast to EAs that generally employ randomization to establish diversification, SS relies on the premise that a systematic design for creating new solutions is generally superior to randomness; and thus SS makes only limited use of randomization. The quality of a particular solution

is determined by its objective value and by its degree of diversity (Resende et al., 2009); cf. (Luke, 2009). The generalized form of SS is called Path Relinking providing unifying principles for recombining solutions based on generalized path constructions (Blum and Roli, 2003). For pseudocodes see (Talbi, 2009; Zäpfel et al., 2010) and for parallel approaches (Alba, 2005).

5.7 Benchmarking

Benchmarking stands for comparing the performance of different solution methods by use of (publicly) available sets of instances of a certain optimization problem, referred to as benchmarks. The use of identical problem instances allows comparing the solution quality gained by different methods in different technical environments (hardware and/or software). When comparing a new metaheuristic to existing ones, it is advantageous to test on the problem instances already tested by previous papers (Silberholz and Golden, 2009).

Comparing metaheuristics may include numerous performance indicators that may be grouped into the following categories: *a*) solution quality, *b*) computational effort, *c*) robustness. *d*) simplicity (Talbi, 2009). Depending on the specific use case, the solution quality (effectiveness) and the computational effort (efficiency) are probably the most important indicators for the performance of (meta)heuristics. Beyond those, in the second line, development cost, ease of use, flexibility (wide applicability) and maintainability may be considered as additional indicators evaluating metaheuristics (Talbi, 2009).

Solution Quality The solution quality of a heuristic is determined by comparing its solution for a certain problem instance with another solution obtained by another method (for the same instance). Basically, there exist four types of solutions compared with each other: *a*) the optimum solution, *b*) a lower/upper bound, *c*) the best known solution, or *d*) the actual implemented solution.

If an optimal solution is available, it is obviously reasonable to use it as the reference. If no optimal solution is available, then lower/upper bounds may serve as reference. And in those cases where neither the global optimum is known nor a lower/upper bound exists, the comparison is performed with the best solution known to this date. The best known solutions may be obtained by other heuristics or by truncated exact methods. Another option is to simply use the solution that is currently implemented in order to determine the quality of the method under study (Talbi, 2009); cf. (Marti and Reinelt, 2011). Refer to Figure 32 that shows the gaps to optimize.

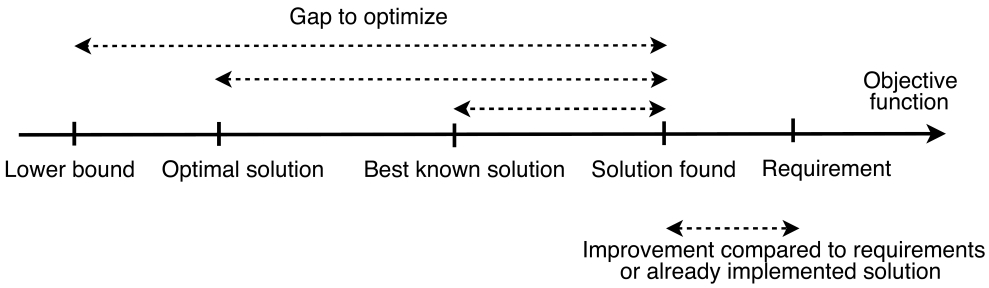


Figure 32: Four types of solutions used for benchmarking (Talbi, 2009)

Computational Effort The computational effort needs to be spend to reach a certain level of solution quality and is the second important performance measure beside the solution quality itself; cf. (Silberholz and Golden, 2009). At the same time, run time comparisons are some of the most difficult comparisons to make. Measuring computation times is critical due to the fact that the measurements depend on the characteristics of the underlying experimental system, e.g. *a*) the hardware involved, *b*) the operating system, *c*) the programming language of choice, and *d*) the compiler (options) used for executing the code. For that reason it might be more suitable in certain situations to use computer system independent performance measures, such as the number of objective function evaluations (Talbi, 2009). Based on computing time measurements it is

possible to experimentally determine the convergence speed of heuristics in order to facilitate the development of heuristics that gain high improvements in very short time. Ovacik and Uzsoy (1995) discuss the trade-off between solution quality and computation time.

Robustness Robustness describes the (in)sensitivity of the obtained solutions to input variable changes that occur after the solution has been found. Robust algorithms lead to stable solutions with low variability for optimization problems with such uncertainties, e.g. changes in input variables slightly modifying the original problem Another view on robustness is particularly concerned with non-deterministic search methods that employ randomness in their strategies (stochastic search). Since the obtained solutions may vary for identical instances, in this context, robustness indicates a low level of deviation over different runs of the algorithm on the same instance (Talbi, 2009). Greenberg and Morrison (2008) investigate problems and strategies in the area of robust optimization more thoroughly.

Simplicity The simplest solution is most often the best. This statement in general holds for the development of software systems and in particular for the design of metaheuristic algorithms. Simpler algorithms are easier to implement, maintain, adapt, explain and analyze (Silberholz and Golden, 2009). Indeed, simplicity results in lower susceptibility to errors. Silberholz and Golden (2009) mention the steps of the pseudocode or the lines of the written code as reasonable metrics, but argues they depend on the used language and the author's style. Another metric describing the complexity of an algorithm is the number of parameters used to set up the method.

5.7.1 Benchmark Results for Example Problems

This section exemplary cites the results reported for three problem types among many others. It can be summarized that among the mentioned benchmark studies no generally superior search scheme emerges. However, with respect to a particular problem class, researchers report about the differences among their tested implementations of common search schemes.

For the $\mathcal{NP}$-hard linear ordering problem (LOP), Marti and Reinelt (2011) and Marti et al. (2012) compare the performance of TS, a memetic algorithm, VNS, SA, SS, GRASP, a GA, and ILS. They observe the best performance for the memetic algorithm, ILS and TS and an acceptable performance for VNS, SS, GRASP, while SA and the GA are classified as poor.

Based on a single-machine scheduling problem, Ibaraki (1997) compares the performance of a GA, genetic LS, multi-start LS, GRASP, SA, and TS. Ibaraki (1997) observes that genetic LS is much more efficient than GA. Further, genetic LS, GRASP, SA, and TS behave more or less similarly and genetic LS and SA perform slightly better than TS and GRASP.

Hansen and Mladenović (2001) present a comparison between GRASP, VNS and TS for a variant of the SAT problem. They observe that VNS and TS perform better than GRASP and TS does slightly better than basic VNS.

Azimi (2004) examine timetabling problems, comparing SA, TS, GAs and ACO with each other. They find that ACO works better on these problems.

Durillo et al. (2010) analyze the convergence speed of seven state-of-the-art metaheuristics for multi-objective benchmark problems. Among those studied methods, one will can find PSO, GA and SS. The results show that PSO is among the most promising approaches to deal with.

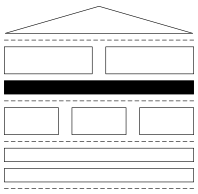
5.7.2 No-Free-Lunch Theorems (NFLTs)

Wolpert and Macready (1997) introduce the NFLTs roughly saying that the average performance of any pair of algorithms across all possible problems is identical, even if one of them is random search. The NFLTs can be visualized with a matrix in which the rows represent problems, the columns stand for search strategies and the entries describe the performance of the strategies over the problems. Basically NFLTs dictate that all rows have the same average (Ho and Pepyne, 2001). Wolpert and Macready (1997) demonstrate the danger of comparing algorithms by their performance on a small sample of problems and emphasize the importance of incorporating problem-specific knowledge into search schemes. Ho and Pepyne (2001) point out that specializing search algorithms to the landscape structure of the focused problem class is the only way one strategy can outperform

another. Many studies on the analysis of landscapes of different optimization problems have shown that not only different problems correspond to different structures but also different instances of the same problem (Talbi, 2009).

5.7.3 Comparative Testing

The dominant procedure in developing new search methods in OR is comparative/competitive testing. A new search strategy or a modification of an existing strategy is implemented and the resulting performance is compared with that of existing algorithms. Most researchers typically focus on demonstrating rather than analyzing algorithm performance in their work (Watson, 2009). As a result, the conditions that cause good or worse solutions often remain unclear. However, the improved algorithm's performance observed in experiments is usually ascribed to the enhancements made to the algorithm. Since there is no proof for this causal relationship, according to Watson (2009), it cannot be ruled out that the observed increase in improvement is probably a result of *a*) the fine tuning of the algorithm or associated parameters, *b*) implementation tricks, or *c*) flaws in the comparative methodology. In the same line Hooker (1995) argues that the strategy of competitive testing lacks deeper insights about cause-effect relationships between an algorithm and its performance. And more clearly, Cohen (1995) states that explaining the performance is better than demonstrating performance.



6 Batch Scheduling

Contents

Scheduling is a decision-making process that deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives (Pinedo, 2005, 2008); cf. (T'kindt and Billaut, 2006) for alternative definitions of the scheduling paradigm.

Generally, batching problems can be seen as a combination of sequencing and partitioning problems. On one hand, the aspect of partitioning notes the task to find a partition of jobs into batches. On the other hand, the aspect of sequencing states the task to find a sequence of batches (Albers and Brucker, 1993). Brucker (2007) defines a parallel batch as a grouped set of jobs jointly processed on the same machine. There exist two types of process batches: serial batches and parallel batches. The abbreviations $s-batch$ for serial batching and $p-batch$ for parallel batching have emerged to widely accepted notions for batching problems in scheduling literature (Brucker, 2007).

A serial process batch refers to a sequence of single jobs that belong to the same process family. The jobs are produced serially (one at a time) on the workstation, which changes over to the next process family after finishing the last job of the current serial batch and before producing the first job of the next serial batch. The concept of serial batching reflects the fact that changing the setup of a workstation from one family to another may take some time, i.e. idle time in which the machine is unavailable for production. The size of a serial batch is naturally unlimited (Brucker, 2007); cf. (Hopp and Spearman, 2001; Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

It is distinguished between bounded and unbounded batching problems within the class of $p-batch$ problems. The maximum size of a batch is commonly denoted with the parameter b, whereas the bounded case is denoted with $b < n$ and $b = \infty$ refers to the unbounded case. The bounded case $b < n$ refers to those batching problems in which the batch size is limited by the machine capacity. The unbounded (easier) case $b = \infty$ does not define a limit on the batch size and usually occurs when the size of the jobs is very small compared to the capacity of the machine. The notion $b = 1$ refers to a conventional scheduling environment without batching, i.e. a machine with unit capacity (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

Typical Use Cases The majority of BPMs performs heat treat operations, simultaneously exposing a number of material items to high temperature in a type of furnace for several hours; cf. (Hopp and Spearman, 2001; Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006).

With the focus on waferfabs, there exist two predominant bounded $p-batch$ scheduling problems: frontend diffusion/oxidation operations and backend burn-in operations. However, bounded $p-batch$ problems can also be found in deposition processes and cleaning operations (Mönch et al., 2011a).

Diffusion and oxidation operations in the waferfab frontends are typical $p-batch$ scheduling problems. The wafers of usually six to twelve jobs jointly undergo a diffusion/oxidation process in a cylindrical reactor. Once the process has been started, no jobs can be removed or added until the processing is completed, i.e. no preemption is allowed. The diffusion/oxidation process is characterized by the existence of incompatible families: jobs that belong to different families, also called recipes, cannot be processed together. The process time of the batch is given by the job

family (recipe). Consequently all jobs of the same family require the same processing time. The batch scheduling problems present in the oxidation/diffusion area usually deal with incompatible job families where all jobs of the same family (recipe) have identical processing times (Mathirajan and Sivakumar, 2006b); cf. (Chandru et al., 1993b; Uzsoy, 1995; Ghosh and Gupta, 1997; Mehta and Uzsoy, 1998).

Another typical $p-batch$ scheduling problem is given by burn-in operations in the waferfab backends. The IC chips are loaded onto boards, placed in an oven and then stressed electrically and thermally in order to force the failure of weak or fragile devices. The burn-operation in the waferfab backends is characterized by the existence of non-identical job sizes. Depending on the product type, the jobs may require a different number of boards that define the size of the job. The capacity of the oven is given by the maximum number of boards it can accommodate. In addition, the boards are often product-specific and therefore of a kind of limited secondary resource. Each IC chip has a minimum burn-in time, and the processing time of the entire batch is given by the longest minimum burn-in time among all the jobs in the batch. Batching jobs of different product types is usually allowed since an IC chip may be longer exposed to the process than specified with the minimum burn-in time. Similar to the diffusion process, preemption is usually not allowed (Mathirajan and Sivakumar, 2006b); cf. (Uzsoy, 1994; Jolai Ghazvini and Dupont, 1998; Kempf et al., 1998).

Scheduling Models Pinedo (2004, 2005, 2008) identifies three types of scheduling problems: *a*) offline deterministic scheduling, *b*) stochastic scheduling, and *c*) onl-ine deterministic scheduling. (Pinedo, 2004, 2005, 2008) first and foremost distinguishes between deterministic scheduling and stochastic scheduling. Deterministic scheduling is then further divided into offline deterministic and online deterministic scheduling problems.

Stochastic scheduling refers to models used to evaluate simple heuristics, e.g. relatively simple priority or dispatching rules, suitable to provide immediate decisions in a real-time control system. This type of scheduling model is also referred to as infinite horizon model, emphasizing the fact that this model type is typically examined with a large number of jobs in the long run; cf. (Neale and Duenyas, 2000; Tajan et al., 2012).

Deterministic scheduling models most often serve to evaluate the performance of optimization methods in solving scheduling problems as a special form of COPs. Most of the research in this area deals with polynomial time algorithms, complexity proofs, exact (usually enumerative) algorithms, worst case analyses, heuristics, and metaheuristics (Pinedo, 2004, 2005, 2008). This type of scheduling model is also referred to as finite horizon model, emphasizing the fact that this model type is a priori limited in its run time by a (small) finite number of jobs indirectly; cf. (Neale and Duenyas, 2000; Tajan et al., 2012).

The offline deterministic scheduling model presupposes that all information regarding the problem is known a priori, i.e. before any decision is made. Given that all the problem data (e.g. the number of jobs, processing times, release dates, due dates, weights, etc.) are known in advance, the decision-making scheduling system seeks for a solution of a COP with the goal to minimize or maximize an objective function.

The online deterministic scheduling paradigm prohibits any knowledge about the scheduling problem at the outset. Neither knowledge about statistical distributions for processing times or job arrivals nor the number of jobs that is going to be released is known in advance. The information is released gradually to the decision-making scheduling system. The arrival of a job becomes known when the job enters the system, that is when the job is presented to the decision-making system. Depending on the model, the processing time either becomes known in the moment the job is released or only when the job is completed (Pinedo, 2004, 2005, 2008).

Another variant of on-line scheduling is semi-online scheduling where some, but not all, information regarding future job releases is known (Pinedo, 2008). This kind of online model is also referred to as look-ahead model, which allows the semi-online algorithm to foresee a given number of jobs before their actual arrivals (Li et al., 2012a).

Pruhs et al. (2004) further examine research in the area of online scheduling models. They define three types: *a*) the online-time paradigm, *b*) the online-time-nonclairvoyance paradigm, and *c*) the online-list paradigm.

Scheduling Methods Dispatching systems set the de facto standard in the scheduling and control of waferfabs (Mönch et al., 2011a); cf. (Leachman, 2002). In the past decades the standard wisdom was that the use of optimization in dispatch applications is technically not feasible. Driven by continuous improvements in computing power and algorithms this has begun to change (Fordyce et al., 2008).

According to Hansen et al. (2009), the growing success of optimization methods is mainly a result of: *a*) the progress in mathematical programming theory and algorithmic design, *b*) the rapid improvements in computer performances, and *c*) a better communication of new ideas and integration in widely used software.

A dispatching system typically employs a set of simple priority rules, which are continuously further developed to complex rule-based decision systems that truly do a reasonable job. However, compared to scheduling systems, dispatching systems based on rules fundamentally lack a robust ability to: *a*) look across time, *b*) look across tools at a tool set, *c*) create an anticipated sequence of events at a tool set over some time horizon, *d*) establish a formal metric, and *e*) search alternatives. Especially the capability of optimization makes scheduling systems superior to dispatching systems, respectively the ability to search alternatives combined with a formal metric in terms of an objective function used to compare alternative decisions (Fordyce et al., 2008).

Most scheduling problems belong to the class of $\mathcal{NP}$-hard problems for which it is widely accepted that no optimal method with polynomial run time exists. The most obvious idea to solve a COP is to just enumerate all feasible solutions. But due to the complexity of most COPs, a simple complete enumeration will result in unacceptable high computing times. The challenge is to develop efficient algorithms that perform better than a simple enumeration (Lee, 2004). See Section 6.3 for a review of exact methods developed to optimally solve BPM scheduling problems. Since it is considered as impossible to optimally solve $\mathcal{NP}$-hard scheduling problems of a practical size in a reasonable time, research with practical background focuses on approximate methods that can lead to feasible schedules with an acceptable quality in considerable less time; cf. (Talbi, 2009).

In contrast to exact methods, approximate methods do not proof the optimality of the obtained solutions. By not having the burden to proof optimality, approximate methods leave parts of the state space unvisited and thus lead to near-optimal solutions in a reasonable time compared to exact algorithms (Dorigo and Stützle, 2004; Talbi, 2009).

Especially for $\mathcal{NP}$-hard problems, exact algorithms perform poor with respect to computing time. Consequently, solving large instances with exact methods is practical impossible, i.e. would take enormous amounts of time to obtain the optimal solution. Approximate methods trade optimality for efficiency (Dorigo and Stützle, 2004); cf. (Talbi, 2009). Approximate methods can be further divided into heuristics and approximation algorithms. See Section 6.4 for a review of approximation algorithms developed for BPM scheduling problems.

Within the class of heuristics, it is basically distinguished between constructive heuristics and search heuristics (Zäpfel et al., 2010); cf. (Talbi, 2009). Construction algorithms describe an incremental procedure: starting from an empty initial solution, they iteratively add solution components until a complete solution is obtained without any backtracking. Constructive heuristics are usually problem-specific, non-iterative, and create one single solution by applying a set of rules based on problem-specific knowledge. See Section 6.5 for a review of constructive heuristics applied to BPM scheduling problems. This is the underlying concept of a typical dispatching system.

Search heuristics follow a certain search scheme that repeatedly examines many different solutions for a given problem in order to find better solutions (Zäpfel et al., 2010); cf. (Talbi, 2009). Search heuristics correspond to metaheuristics in a broader sense. Informally, a metaheuristic states an algorithmic advancement of a simple heuristic, which is commonly defined as a rule of thumb that leads to near-optimal solutions without complete knowledge of the problem. In contrast to exact methods, metaheuristics lead to acceptable solutions in a reasonable time: solution quality and computing time is generally not exactly defined, i.e. acceptable and reasonable (Talbi, 2009). Metaheuristics primarily justify their use with a well-balanced performance characteristic that describes a favorable trade-off between solution quality and computing time. See Section 6.6 for a review of metaheuristics applied to BPM scheduling problems.

Reviews Leung (2004), Pinedo (2005, 2008), T'kindt and Billaut (2006) and Brucker (2007) discuss the theory of scheduling profoundly. They give detailed explanations of scheduling algorithms

and their applications in manufacturing. Potts and Strusevich (2009) explore the historical developments in scheduling theory, reviewing 50 years of scheduling while highlighting the main topics of scheduling research.

Uzsoy et al. (1992b, 1994) review research about shop-floor control and production planning in semiconductor industry. Gupta et al. (2006) explore the role of operational planning and control of semiconductor wafer production. More recently, Mönch et al. (2009, 2011a) review problems and solution techniques in scheduling waferfab operations and point out future challenges.

Potts and van Wassenhove (1992) and Webster and Baker (1995) provide the first reviews about batching problems, discussing early research in the field of batch scheduling theory with focus on single-machine scheduling models. Potts and Kovalyov (2000) give an extended review of batching problems in various machine environments and for several objectives, putting emphasis on complexity results as well as on the efficiency and effectiveness of algorithms. Mathirajan and Sivakumar (2003) especially review batch scheduling problems in semiconductor manufacturing, classifying batching problems into 12 groups while distinguishing between stochastic and deterministic problems. They refine their classification schemes and systematically organize the published articles in an updated survey three years later (Mathirajan and Sivakumar, 2006b). Mönch et al. (2011a) briefly survey parallel batching problems within a discussion about problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. Beyond semiconductor industries, Méndez et al. (2006) discuss scheduling problems in batch plants, e.g. in a polymer batch plant and a steel-making casting plant.

The following publications do not claim to provide a review but do review research related to their works. Perez et al. (2005) present related works about batching problems with and without incompatible families in different machine environments, i.e. single and parallel environments among others. Koh et al. (2005) concentrate on BP problems with incompatible families and different sizes in a single machine environment. Damodaran and Vélez-Gallego (2012) put emphasis on parallel machine problems, considering incompatible families and different sizes in their review. Chung and Jang (2009) and Gokhale and Mathirajan (2011, 2014) list batching problems in different machine environments, considering unequal ready times and job sizes. Ozturk et al. (2012) present an overview on single and parallel machine problems with and without release dates.

6.1 Scheduling and Control Systems

Depending on the focused time horizon, decision-making in manufacturing is generally organized in three types of activities: *a*) planning for long terms (strategy), *b*) scheduling for medium terms (tactic), and *c*) control for short terms (operation) (Hopp and Spearman, 2001); cf. (Uzsoy et al., 1992b; Pinedo, 2005, 2008; Foote and Murty, 2008).

The exact definition of the time scale between long, medium and short terms differs depending on the type of manufacturing. For a high-tech industry such as semiconductor industry it is reasonable to define *a*) weeks as the long-term time scale (planning), *b*) hours as the middle-term time scale (scheduling), and *c*) seconds as the short-term time scale (control). Based on this refinement, control is concerned with immediate actions in real time, also referred to as real-time control (Monfared and Yang, 2007); cf. (Hopp and Spearman, 2001). Refer to Figure 33 that shows the time horizons for planning, scheduling and control functions.

Beyond the time horizon criterion and connected with it, uncertainty is another index of differentiation between planning, scheduling and control. It is quite reasonable that uncertainty in decisions increases with an increasing time horizon. Hence, control decisions are more certain than the scheduling and planning decisions (Monfared and Yang, 2007). Méndez et al. (2006) note that a critical issue is to guarantee consistency and optimality between these different decision levels.

The scope of this work excludes planning issues related to long terms and focus on scheduling and control for medium and short terms with an emphasis on scheduling powered by optimization. In particular this work examines the $p-batch$ scheduling problem in the diffusion/oxidation area addressed in a parallel machines environment. The focused problem is described in Section 1.2.

At any point in time, the decision-making system faces a COP to solve, which basically involves three activities: *a*) batching, *b*) sequencing, and *c*) partitioning. The batching decision itself is a partitioning problem, referring to the problem of finding groups of jobs to be processed together as a batch. The sequencing problem asks for an optimal processing order of the jobs. In the case of

minimum two idle machines, the sequencing problem is accompanied with a partitioning problem asking for a proper mapping between the jobs and the machines. The COP in form of a scheduling problem requires a solution in terms of a feasible schedule, i.e. a sequence of batches partitioned to the machines, so that the schedule to be executed is optimal with respect to a defined objective function.

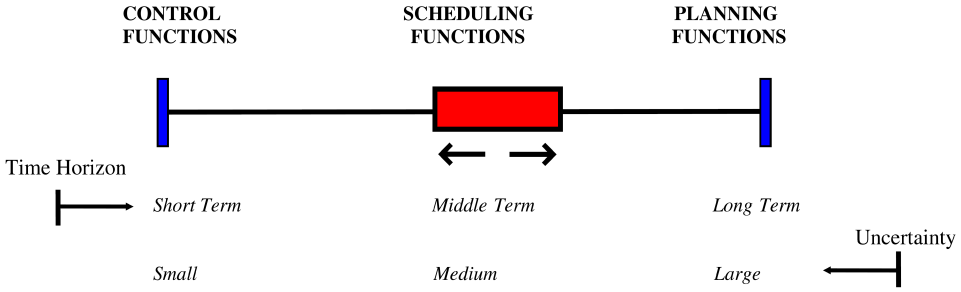


Figure 33: Time horizons for planning, scheduling and control functions (Monfared and Yang, 2007)

Real-Time Control Systems A real-time control system continuously senses the system state variables, deriving proper activities in real-time in order to keep the system running as desired (Monfared and Yang, 2007). Decision-making and subsequent activity is either triggered by events or by repeatedly checking the status of the system (polling), e.g. the machine status. Typically more than one machine is available for processing each job, i.e. a set of parallel machines.

The standard strategy in waferfab dispatching systems is to use a round-robin procedure that controls the machines. It breaks down the scheduling problem for a number of parallel machines to a simpler single machine sequencing problem. The round-robin procedure cyclically checks idleness of the machines and repeatedly selects the next job for processing according to an ordered list of jobs as a result of a priority-based rule. Whenever a machine is ready for processing again, the top-priority jobs are assigned to it.

The sequencing problem is tackled with priority-rules creating an ordered list of jobs sorted by some defined job attributes (e.g. due date). The batching problem is most often indirectly solved as a result of the sequencing decision in a sense that the top priority job defines the next batch to process. Partitioning in terms of mapping jobs/batches to machines is basically an effect of randomness and not explicitly controlled. A top-priority job in the first place of an ordered list is simply assigned to the machine that first becomes free, i.e. first completes its current process. If two idle machines are available, the round-robin procedure will determine which machine is to choose. Refer to Figure 34 depicting an illustration of a working real-time control system.

Scheduling Systems The classical approach to scheduling comprises two activities in succession: constructing the schedule and executing the schedule. The schedule is constructed before a work period begins and then is automatically executed in the real world during that period. This approach is also referred to as "schedule, then execute" approach. The scheduling system is basically composed of two independently acting sub-systems: The scheduling system that repeatedly generates an optimized schedule and the execution system that derives activities from the schedule in order to implement the schedule on the shop floor. Whenever a decision has to be taken, i.e. a machine completes a process and is ready for production again, the upcoming decision is derived from the schedule computed before. This approach is reasonable on a coarse time scale, since reports from industries such as semiconductor fabrication estimate the effective useful life of a schedule as an hour or less.

As time proceeds the shop floor state continuously changes with occurring events, resulting in deviations between the initial schedule and the actual state of the shop floor. These deviations make it impossible to execute the initial schedule any further. When applying a scheduling system on the shop floor, the scheduling system repeatedly solves a deterministic scheduling problem

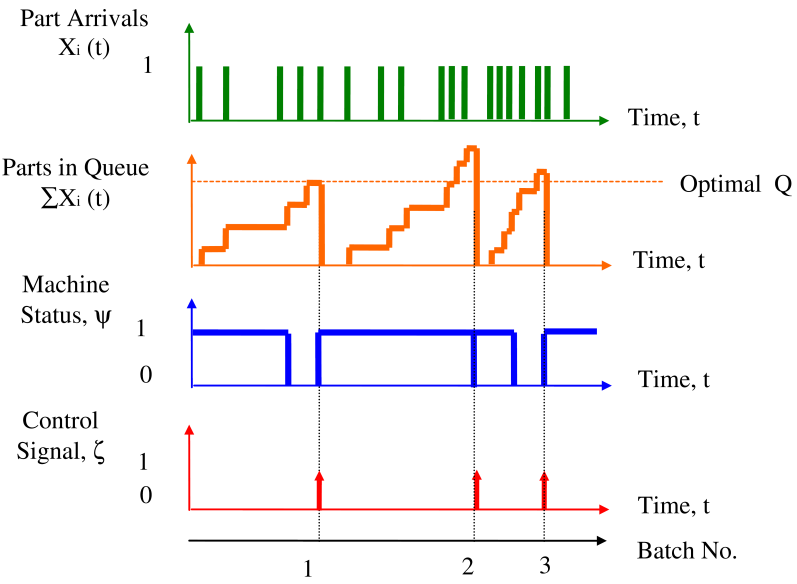


Figure 34: Illustration of a working real-time control system (Monfared and Yang, 2007)

in order to constantly provide a feasible (up-to-date) schedule to be executed on the shop floor (van Dyke Parunak, 1995). Refer to Figure 35 that visualizes the rolling horizon philosophy for periodical rescheduling.

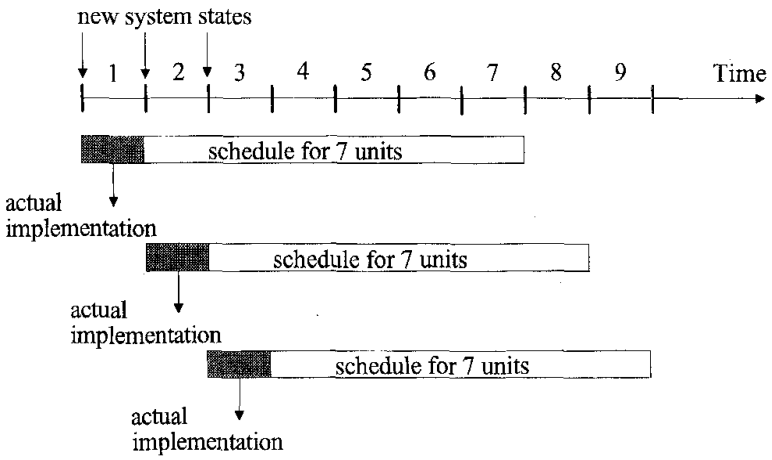


Figure 35: The rolling horizon philosophy for periodical rescheduling (Liao et al., 1996)

Rescheduling In an ideal scheduling environment the factory operations on the shop-floor follow a valid and optimal schedule at any time. But in the real-world an optimal schedule is likely to loose optimality or even becomes infeasible after it is constructed (i.e. when it gets online) due to the fact that the manufacturing system changes continuously as time proceeds.

Monfared and Yang (2007) give examples of disturbances that trigger control, scheduling and planning activities. Events that disturb control decisions may relate to late material arrival, varying process times, or machine breakdowns among others. Events that detract schedules may refer to new job arrivals, changes in due dates or priorities among others. Noack et al. (2011) analyze the frequency of events in a 200 mm high-mix waferfab, e.g. reporting that an unscheduled machine breakdown occurs every 6.6 minutes.

The theoretical approach to scheduling is often not directly applicable due to the dynamic characteristics of the actual situation: its applicability is limited to those situations that are fundamentally static and behave exactly like the models (McKay et al., 1988).

Given that the period of time between generating and executing the schedule is not negligible, the probability to process a schedule as exactly as planned is very low in the moment the schedule has to be applied The quality of a schedule is valid before the schedule becomes online, but then the problem is to maintain a feasible solution with a good quality. In such a context, it is required to modify the proposed schedule in order to permanently provide a feasible solution at any time. This is due to various sources of uncertainty we face in a real-world context, e.g. machines can break down or processing times are not perfectly known, among many others (T'kindt and Billaut, 2006).

For the reason of simplication when creating a schedule, it is assumed that the situation on the shop oor will remain unchanged throughout the entire time horizon. However, industrial environments are typically highly dynamic and the proposed initial schedule can quickly become infeasible after the occurrence of unforeseen events, e.g. machine breakdown/startup, changes in resource availabilities, or changes in processing/setup times among many others (Méndez et al., 2006).

Whenever a schedule becomes infeasible, the schedule needs to be constructed again in order to maintain schedule feasibility. Rescheduling should be performed in a smooth way in a sense that two consecutively constructed schedules share similarities in order to keep the impact of periodically changing schedules on the shop floor manageable (Liao et al., 1996).

A certain level of stability and continuity improves building up confidence in the schedules constructed for execution. A strategy to avoid time-expensive, full-scale rescheduling is to allow only limited changes to the scheduling decisions already made at the beginning of the time horizon (Méndez et al., 2006).

Rescheduling approaches in complex production systems address aspects of stability and real-time control in order to regain feasibility and quality of deviated schedules in real-time without affecting coordination problems on the shop floor (Dangelmaier et al., 2006, 2007). Refer to Figure 36 that visualizes the real-world state evolution and interaction with the rescheduling system.

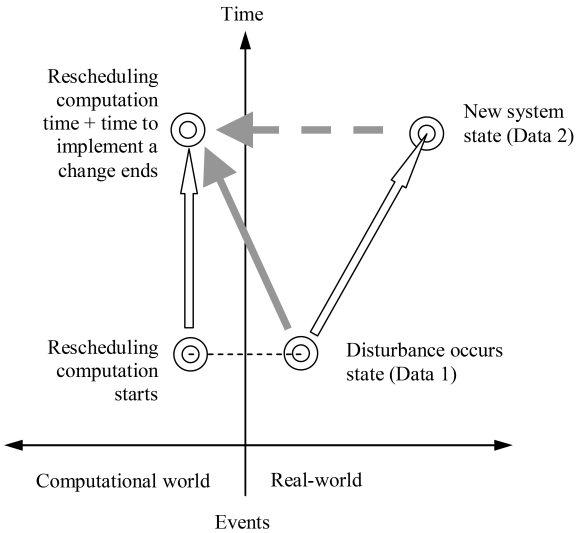


Figure 36: Real-world state evolution and interaction with the rescheduling system (Dangelmaier et al., 2007)

Robust Optimization It is a matter of fact that the quality of a schedule is valid before the schedule becomes online and the problem is to maintain a feasible solution with a good quality throughout its online status. A non-feasible schedule can be modified very quickly in real-time in order to regain feasibility. But if the initial schedule is very sensitive to changes, the quality of the

schedule may deteriorate remarkably. It is highly desired to construct robust schedules that are not too sensitive to changes with respect to quality. In this context, the decision-maker tends to search for a compromise between the quality and the robustness of a schedule.

The classical scheduling approach that does not take uncertainty into account and only aims on quality is also referred to as a predictive scheduling approach. Scheduling approaches that explicitly deal with uncertainties either belong to the class of proactive or reactive approaches. Proactive approaches take knowledge of uncertainty into account when constructing a schedule. Reactive approaches revise the schedule in real-time whenever the schedule is subject to unexpected events (T'kindt and Billaut, 2006).

A relevant group in the research area that deals with optimization problems including uncertainty in their models corresponds to the class of SCOPs. Bianchi et al. (2009) survey SCOPs that deal with partially unknown problem data, i.e. with probability distributions as input data. Robust optimization refers to problem solving with uncertainties, taking into account that a solution remains acceptable after slight changes of the decision variable values (Talbi, 2009). Greenberg and Morrison (2008) discuss robust optimization models, in particular the mean-risk model, the recourse model, and the chance-constrained model.

6.2 Complexity Review

A significant amount of research in the area of complexity theory focuses on boundaries of polynomial time and $\mathcal{NP}$-hard problems. Scheduling theory seeks to define the borderline between the hardest or the most general problems that still can be solved in polynomial time and the simplest or least general problems that are $\mathcal{NP}$-hard, either in the ordinary sense or strongly (i.e. under binary or unary encoding) (Pinedo, 2008). The table in Appendix B provides an overview on the entire set of $p - batch$ problem variants and their complexity status.

Timkovsky (2004) identifies six complexity status results: a scheduling problem is either *a*) polynomially solvable (is in $\mathcal{P}$), *b*) pseudo-polynomially solvable, but neither polynomial solvability nor $\mathcal{NP}$-hardness is established, *c*) $\mathcal{NP}$-hard, but neither pseudo-polynomial solvability nor strong $\mathcal{NP}$-hardness is established, *d*) ordinarily $\mathcal{NP}$-hard, i.e. pseudo-polynomially solvable and $\mathcal{NP}$-hard, *e*) strongly $\mathcal{NP}$-hard, or *f*) the complexity status is open.

Generally, complexity results of scheduling problems presume that the number of machines is fixed. Based on this assumption, a polynomial algorithm runs in polynomial time with respect to the number of jobs but is not necessarily polynomial in the number of machines. If a problem is said to be $\mathcal{NP}$-hard, either in the ordinary sense or strongly, then $\mathcal{NP}$-hardness holds for a fixed number of machines (Pinedo, 2008).

It is shown that slight modifications of a (batch) scheduling problem may turn $\mathcal{NP}$-hardness into polynomial solvability or vice versa (Albers and Brucker, 1993). There exist some common simplifications in form of additional constraints that change the structure of a problem, which in turn may change its complexity status. A scheduling problem in its general, unrestricted form allows for arbitrarily set processing times, weights, due dates, release dates and other variables. But processing times as well as other variables may be restricted, for example, in a sense that all jobs have the same processing time. Identical processing times are then denoted with $p_j = p$. Similarly, it may be required that all jobs have the same weight ($w_j = w$), due date ($d_j = d$), deadline ($\bar{d}_j = \bar{d}$), and/or release date ($r_j = r$). In these cases, it is said that the variable is no longer arbitrary but common to all jobs. Another simplification is given by the constraint of agreeable variables. For example, a scheduling problem with agreeable processing times and due dates implies that the job with the smallest processing time also shows the smallest due date value, i.e. $p_i < p_j$ implies that $d_i < d_j$ $(1 \leq i < j \leq n)$. These simplifications often have strong influence on the complexity of a problem; cf. (Lawler, 1977; Graham et al., 1979; Leung, 2004; Pinedo, 2005, 2008).

Pinedo (2008) also states that models that are $\mathcal{NP}$-hard in a deterministic setting often allow a simple priority policy to be optimal in a stochastic setting.

Landmark Publications in Complexity Theory for Scheduling Important cornerstones in complexity theory research and a remarkable number of complexity results known today is based on the works of Lenstra et al. (1977), Lawler (1977), Garey and Johnson (1978) and Graham et al.

(1979) who present landmark results in this field. Lenstra et al. (1977) are one of the first ones who examine the complexity status of machine scheduling problems, presenting polynomial bounded algorithms or establishing unary/binary $\mathcal{NP}$-completeness for a number of basic scheduling problems. In the same vein Lawler (1977) studies complexity results with focus on the encoding scheme, presenting a pseudo-polynomial algorithm for sequencing jobs to minimize TT on a single machine with unit capacity. Garey and Johnson (1978) provide a standard framework for stating and proving strong $\mathcal{NP}$-completeness results. Graham et al. (1979) give the first reviews on deterministic scheduling problems, examining several basic problems and their complexity status.

Lageweg et al. (1982) describe a computer program that provides a record of the known complexity results machine scheduling problems as a result of a listing of complexity results and reduction rules.

Scheduling Complexity Reviews Potts and Strusevich (2009) review major research achievements in the first 50 years of scheduling, providing a chronological review of the most important works in the development of scheduling algorithms. Chen et al. (1999) provide a detailed review of machine scheduling, summarizing complexity results and algorithms for many scheduling problems. Comprehensive studies in the area of scheduling in general can be found in (Leung, 2004; Pinedo, 2005, 2008; T'kindt and Billaut, 2006). Brucker and Knust (2014) maintain a continuously updated webpage[7] with complexity results of the most common machine scheduling problems.

Potts and van Wassenhove (1992) are one of the first ones who provide a review especially for scheduling problems with a focus on complexity results. Nearly at the same time, Albers and Brucker (1993) examine the complexity of one-machine batching problems. Potts and Kovalyov (2000) provide a more recent review on complexity results for batch scheduling problems. Liu (2007) studies many variants of bounded and unbounded batch scheduling problems with focus on complexity results, also summarizing time complexities for bi-criterion scheduling on a single batch machine.

6.2.1 Complexity Hierarchies

Reduction rules provide an important methodical vehicle in determining the complexity status of scheduling problems, i.e. by reducing a scheduling problem to another for which the complexity status is known. Graham et al. (1979) originally present such reduction rules for scheduling problems.

Given that an algorithm for one scheduling problem can be applied to another scheduling problem as well, it is possible to derive complexity results between different problems. For example, $1 \mid\mid \sum C_j$ is a special case of $1 \mid\mid \sum w_j C_j$ and a procedure for $1 \mid\mid \sum w_j C_j$ can clearly also be used to solve $1 \mid\mid \sum C_j$. In complexity terminology it is then said that $1 \mid\mid \sum C_j$ reduces to $1 \mid\mid \sum w_j C_j$, usually denoted by $1 \mid\mid \sum C_j \propto 1 \mid\mid \sum w_j C_j$. Based on this concept a chain of reductions between some scheduling problems can be established. However, there exist many problems that are not comparable with one another, underpinned by the fact that no suitable reduction rules exist that can be used to map a problem to another (Pinedo, 2005, 2008). Obviously, a $p - batch$ scheduling problem with batch size identical to one is equivalent to the corresponding standard single machine problem with unit capacity. Thus $\mathcal{NP}$-hardness results for single machine problems are also valid for $p - batch$ problems with bounded batch size (Brucker, 2007). Refer to Figure 37 showing the basic reduction rules that are most important.

Unfortunately, there exist quite similar $p - batch$ scheduling problems that are not reducible to each other. This might lead to some confusions. The reason is that we face different processing time models: the longest job processing time model and the family processing time model. The longest job processing time model describes the case in which the jobs have arbitrary processing times and the processing time of the batch is determined by the longest processing time of its jobs. This kind of model is originally motivated by the backend burn-in operations. The family processing time model is usually motivated by diffusion/oxidation operations that deal with incompatible job families. In this case, the processing time of a batch is given by the corresponding job family and different job families have different processing times.

[7] http://www.informatik.uni-osnabrueck.de/knust/class/

Let us consider a few examples to make the difference clear. Based on the longest job processing time model, Brucker et al. (1998) show that $1 \mid p-batch, b<n, r_j \mid C_{max}$ is unary $\mathcal{NP}$-hard. Yuan et al. (2004) also assume that the processing time of a batch is equal to the maximum processing time among the jobs in the batch, but additionally incorporates incompatible job families. They proof that $1 \mid p-batch, b<n, r_j, fmls \mid C_{max}$ is strongly $\mathcal{NP}$-hard, even for the unbounded case. In contrast, Uzsoy (1995) presents a polynomial algorithm for $1 \mid p-batch, b<n, r_j, fmls \mid C_{max}$. The difference is that Uzsoy (1995) employs the family processing time model in which the processing time is equal for all jobs of the same family.

Another difference in complexities arises with the L_{max} objective. Brucker et al. (1998) show that $1 \mid p-batch, b<n \mid L_{max}$ is unary $\mathcal{NP}$-hard whereas $1 \mid p-batch, b<n, fmls \mid L_{max}$ is solvable in polynomial time (Uzsoy, 1995).

Similar observations are made for the TT and TU objectives. By reduction to $1 \mid p-batch, b<n \mid L_{max}$, it is deduced that $1 \mid p-batch, b<n \mid \sum U_j$ and $1 \mid p-batch, b<n \mid \sum T_j$ are unary $\mathcal{NP}$-hard, given that the longest job processing time model is applied. With respect to the family processing model, unary $\mathcal{NP}$-hardness is established for $1 \mid p-batch, b<n, fmls \mid \sum U_j$ in (Liu and Zhang, 2008) and for $1 \mid p-batch, b<n, fmls \mid \sum T_j$ in (Mehta and Uzsoy, 1998).

Consequently, the complexity results based on family processing time models are not transferable to problems that use the longest job processing time model and vice versa. The results indicate that family processing times simplify the scheduling problem compared to the longest processing time model.

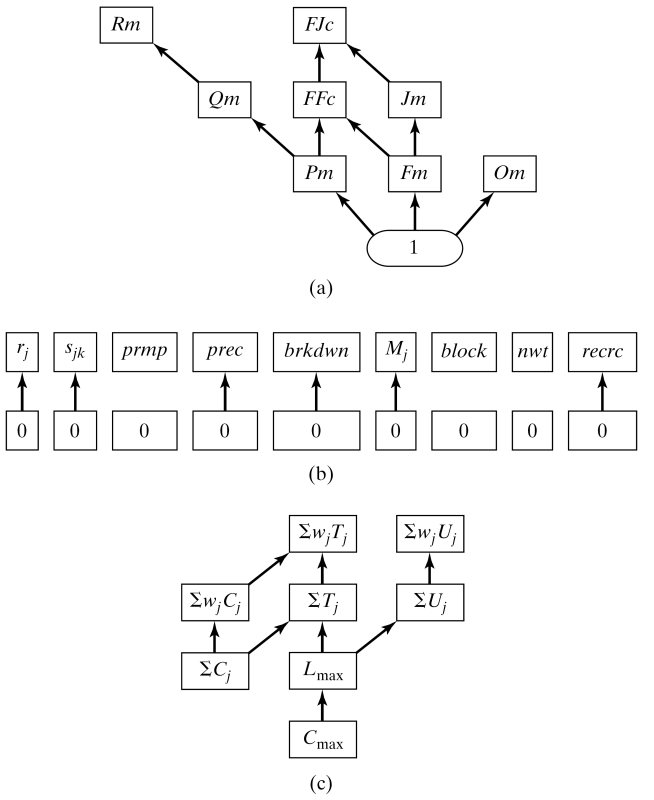


Figure 37: Complexity hierarchies (Pinedo, 2008) (cf. (Graham et al., 1979))

6.2.2 Makespan

Brucker et al. (1998) and Sung and Choung (2000) show that $1 \mid p-batch, b<n \mid C_{max}$ can be solved optimally in polynomial time; cf. (Brucker, 2007).

Release Dates The existence of release dates makes the problem harder to solve. Brucker et al. (1998) proof that $1 \mid p-batch, b < n, r_j \mid C_{max}$ is unary $\mathcal{NP}$-hard. They show that $1 \mid p-batch, b < n, r_j \mid C_{max}$ is an equivalent mirror image problem of $1 \mid p-batch, b < n \mid L_{max}$, which is shown to be $\mathcal{NP}$-hard in the strong sense even if $b = 2$ based upon a reduction from the unary $\mathcal{NP}$-complete problem 3-Partition; cf. (Brucker, 2007). Liu and Yu (2000) proof that $1 \mid p-batch, b < n, r_j \mid C_{max}$ is binary $\mathcal{NP}$-hard even for the case with two distinct release dates by reduction to the Partition problem.

Liu and Yu (2000), Poon and Zhang (2000) and Deng et al. (2003) present pseudo-polynomial algorithms for the case with distinct release dates, i.e. a fixed number of release dates. Poon and Zhang (2000) additionally consider the case with distinct processing times and distinct release times. Sung et al. (2002) present a DP algorithm with polynomial run time when the number of processing times is fixed. Also Lee and Uzsoy (1999) present polynomial and pseudo-polynomial algorithms for several special cases of $1 \mid p-batch, b < n, r_j \mid C_{max}$.

The simple First-Only-Empty (FOE) algorithm (Ikura and Gimple, 1986) is optimal for the case with equal processing times, i.e. the process time of a batch is constant and independent of the batch size. The idea of this algorithm is to process the jobs by making only the first batch partially empty; the rest of batches are full, which results in a minimum number of batches. FOE solves $1 \mid p-batch, b < n, r_j, p_j = p \mid C_{max}$ in polynomial run time. Additionally, Ikura and Gimple (1986) present a polynomial algorithm for the same problem with deadlines where the release dates deadlines are agreeable $(1 \mid p-batch, b < n, r_j, \bar{d}_j, p_j = p, agreeable(r_j, \bar{d}_j) \mid C_{max})$

However, the unbounded $p-batch$ scheduling problem is easier to solve. There exist polynomial algorithms for the unbounded problem $1 \mid p-batch, r_j \mid C_{max}$ (Poon and Zhang, 2000; Liu, 2007). Furthermore, Liu et al. (2003) establish the pseudo-polynomial solvability of the unbounded batch machine scheduling problem with job release dates and any regular objective, i.e. $1 \mid p-batch, r_j \mid \cdot$ is pseudo-polynomially solvable for any regular objective.

If the (bounded) problem is subject to incompatible families, i.e. relies on the family processing time model, then the problem also looses $\mathcal{NP}$-hardness; Uzsoy (1995) shows that the problem $1 \mid p-batch, b < n, r_j, fmls \mid C_{max}$ can be solved polynomially.

Non-Identical Job Sizes Similar to release dates, the existence of non-identical job sizes makes the problem harder to solve. Uzsoy (1994) shows that $1 \mid p-batch, B, s_j \mid C_{max}$ is strongly $\mathcal{NP}$-hard, since the problem with identical processing times is equivalent to the bin packing problem, which is strongly $\mathcal{NP}$-hard. This complexity result is very important within the scope of this complexity review since the $\mathcal{NP}$-hardness proof is also valid for the case with constant processing times. This means that this complexity result also holds for the other batch scheduling problems with non-identical job sizes, in particular for those based on the family processing time model.

Incompatible Job Families Given that the problem relies on the family processing time model, the existence of incompatible families makes the problem easier. Uzsoy (1995) shows that the problem $1 \mid p-batch, b < n, fmls \mid C_{max}$ and the related version with job arrivals $(1 \mid p-batch, b < n, r_j, fmls \mid C_{max})$ can be solved polynomially.

However, Yuan et al. (2004) establish strong $\mathcal{NP}$-hardness for the unbounded single machine parallel batch scheduling problem with incompatible family jobs and release dates to minimize C_{max} when the the processing time of a batch is equal to the maximum processing time among the jobs in the batch. Thus Yuan et al. (2004) also proof that the bounded case $1 \mid p-batch, b < n, fmls \mid C_{max}$ is strongly $\mathcal{NP}$-hard with respect to the longest-processing time model.

Parallel Machines Garey and Johnson (1978) show that $P \mid\mid C_{max}$ is $\mathcal{NP}$-hard in the strong sense, which implies that the corresponding bounded batching problem $P \mid p-batch, b < n \mid C_{max}$ is at least as hard to solve. The proof for the complexity of $P \mid\mid C_{max}$ is given by a reduction from the 3-Partition problem (Leung, 2004) The strongly $\mathcal{NP}$-hard problem becomes $\mathcal{NP}$-hard in the ordinary sense when the number of parallel machines is limited to two at maximum, since $P2 \mid\mid C_{max}$ is reducible to the Partition problem (Leung, 2004; Pinedo, 2008).

6.2.3 Maximum Lateness

Finding whether there is a feasible solution to the bounded problem in which the jobs have deadlines is $\mathcal{NP}$-complete in the strong sense even if $b = 2$. The proof is based upon a reduction from the unary $\mathcal{NP}$-complete problem 3-Partition (Brucker et al., 1998); cf. (Brucker, 2007). This implies that $1 \mid p - batch, b < n \mid L_{max}$ is unary $\mathcal{NP}$-hard.

But there exist polynomial algorithms for some special cases. Lee et al. (1992) show that the problem $1 \mid p - batch, b < n, p_j = p \mid L_{max}$ can be solved polynomially. In this case a full batch schedule leads to the optimum when the jobs are sequenced in order of smallest to largest due date (Webster and Baker, 1995). Lee et al. (1992) also present a polynomial algorithm for the problem with agreeable processing times and due dates. In the same vein the unbounded model is in $\mathcal{P}$ (Brucker et al., 1998); (Brucker, 2007)

Release Dates Cheng et al. (2001) show that the unbounded case $1 \mid p - batch, r_j \mid L_{max}$ is ordinary $\mathcal{NP}$-hard; cf. (Liu, 2007). The problem $1 \mid p - batch, r_j \mid L_{max}$ is proved to be $\mathcal{NP}$-hard under the binary encoding even if the processing times and deadlines are agreeable. The proof is based on a reduction from the Partition problem. However, its complexity under the unary encoding scheme remains open. Cheng et al. (2001) propose polynomial algorithms for several special cases of $1 \mid p - batch, r_j \mid L_{max}$. Furthermore, Liu et al. (2003) establish pseudo-polynomial solvability for $1 \mid p - batch, r_j \mid \cdot$ with any regular objective.

The bounded case $1 \mid p - batch, b < n, r_j \mid L_{max}$ is unary $\mathcal{NP}$-hard, since Lenstra et al. (1977) proof strong $\mathcal{NP}$-hardness for the corresponding unit capacity problem $1 \mid r_j \mid L_{max}$; cf. (Pinedo, 2008). It can be said that $1 \mid r_j \mid L_{max} \propto 1 \mid p - batch, b < n, r_j \mid L_{max}$. Another variant of reduction is given by the complexity result for minimizing C_{max}. As stated before, Brucker et al. (1998) establish strong $\mathcal{NP}$-hardness for $1 \mid p - batch, b < n, r_j \mid C_{max}$; cf. (Brucker, 2007). Given that minimizing C_{max} is easier than minimizing L_{max}, it is deduced that $1 \mid p - batch, b < n, r_j \mid C_{max} \propto 1 \mid p - batch, b < n, r_j \mid L_{max}$. Li and Lee (1997) proof that $1 \mid p - batch, b < n, r_j \mid L_{max}$ is strongly $\mathcal{NP}$-hard even with agreeable release times and due dates. They show that the problem is solvable in polynomial time when the processing times are also agreeable with the release times and due dates of the jobs. Polynomial solvability is also achieved when the processing times are constant (Webster and Baker, 1995; Baptiste, 2000; Cheng et al., 2001). Similarly, Lee et al. (1992) present a polynomial algorithm for constant processing times in conjunction with agreeable release dates and due dates.

Non-Identical Job Sizes The problem $1 \mid p - batch, B, s_j \mid L_{max}$ is strongly $\mathcal{NP}$-hard. Uzsoy (1994) show that $1 \mid p - batch, B, s_j \mid C_{max}$ is strongly $\mathcal{NP}$-hard, since the problem with identical processing times is equivalent to the strongly $\mathcal{NP}$-hard bin packing problem. Applying the reduction rules, it is deduced that $1 \mid p - batch, B, s_j \mid C_{max} \propto 1 \mid p - batch, B, s_j \mid L_{max}$.

Incompatible Job Families Uzsoy (1995) describes a polynomial algorithm for $1 \mid p - batch, b < n, fmls \mid L_{max}$, assuming identical processing times of jobs of the same family. This is in contrast to the complexity result for $1 \mid p - batch, b < n \mid L_{max}$ based on the longest processing time model for which Brucker et al. (1998) proof strong $\mathcal{NP}$-hardness; cf. (Brucker, 2007).

Parallel Machines $P \mid p - batch, b < n \mid L_{max}$ is strongly $\mathcal{NP}$-hard by reduction to $P \mid\mid C_{max}$ for which Garey and Johnson (1978) proof $\mathcal{NP}$-hardness; (Lee et al., 1992). Brucker et al. (1998) proof strong $\mathcal{NP}$-hardness for $1 \mid p - batch, b < n \mid L_{max}$ that also reduces to $P \mid p - batch, b < n \mid L_{max}$. Lin and Jeng (2004) present a DP algorithm with pseudo-polynomial run time The computational complexities become pseudo-polynomial when the number of machines is fixed. Liu (2007) and Liu et al. (2009) examine the complexity status for the problem of minimizing L_{max} on parallel unbounded BPM. It is proven that minimizing any due date related objective on parallel unbounded BPM is strongly $\mathcal{NP}$-hard (Liu, 2007) even if the release dates and deadlines are agreeable (Liu et al., 2009)

6.2.4 Tardiness

The problem $1 \mid p\text{–}batch, b < n \mid \sum T_j$ is strongly $\mathcal{NP}$-hard by reduction to $1 \mid p\text{–}batch, b < n \mid L_{max}$ for which Brucker et al. (1998) present the complexity proof based upon a reduction from the unary $\mathcal{NP}$-complete problem 3-Partition; cf. (Brucker, 2007). Obviously the weighted case $1 \mid p - batch, b < n \mid \sum w_j T_j$ is then strongly $\mathcal{NP}$-hard, too.

In the case of agreeable processing times and due dates, Liu (2007) proves that the problem $1 \mid p - batch, b < n \mid \sum T_j$ is binary $\mathcal{NP}$-hard even if the machine capacity is two. Brucker (2007) presents a polynomial DP algorithm for the case with identical processing times $1 \mid p - batch, b < n, p_j = p \mid \sum T_j$. Liu (2007) further shows that minimizing $1 \mid p - batch, b < n \mid \sum w_j T_j$ remains strongly $\mathcal{NP}$-hard when processing times and due dates are agreeable.

It is interesting to know that even the related unit capacity machine problems are $\mathcal{NP}$-hard as well as the corresponding unbounded batch problems. Lawler (1977) proofs unary $\mathcal{NP}$-hardness for $1 \mid\mid \sum w_j T_j$. The unweighted variant $1 \mid\mid \sum T_j$ is still $\mathcal{NP}$-hard in the ordinary sense, i.e. binary $\mathcal{NP}$-hard, but pseudo-polynomially solvable (Leung, 2004; Pinedo, 2008). The unbounded problem $1 \mid p - batch \mid \sum T_j$ is ordinary $\mathcal{NP}$-hard, i.e. binary $\mathcal{NP}$-hard (Liu et al., 2003) and pseudo-polynomially solvable (Brucker et al., 1998). In the same vein, $1 \mid p - batch \mid \sum w_j T_j$ is ordinary $\mathcal{NP}$-hard, i.e. binary $\mathcal{NP}$-hard (Brucker et al., 1998) by reduction from the Partition problem but not strongly $\mathcal{NP}$-hard since (Baptiste et al., 2004) show that a pseudo-polynomial algorithm exists.

Release Dates The problem $1 \mid p - batch, b < n, r_j \mid \sum T_j$ is unary $\mathcal{NP}$-hard, since Lenstra et al. (1977) proof strong $\mathcal{NP}$-hardness for the corresponding unit capacity problem $1 \mid r_j \mid L_{max}$. A chain of reductions may have the form $1 \mid r_j \mid L_{max} \propto 1 \mid p - batch, b < n, r_j \mid L_{max} \propto 1 \mid p - batch, b < n, r_j \mid \sum T_j \propto 1 \mid p - batch, b < n, r_j \mid \sum w_j T_j$.

The problem looses $\mathcal{NP}$-hardness when the processing times are restricted to be equal; minimizing $\sum T_j$ with identical processing times and release dates $(1 \mid p - batch, b < n, r_j \mid \sum T_j)$ can be solved polynomially (Baptiste, 2000; Brucker, 2007).

However, the unbounded cases $1 \mid p - batch, r_j \mid \sum T_j$ and $1 \mid p - batch, r_j \mid \sum w_j T_j$ share ordinary $\mathcal{NP}$-hardness; cf. (Liu, 2007). On one hand, Cheng et al. (2001) proof binary $\mathcal{NP}$-hardness for the easier problem $1 \mid p - batch, r_j \mid L_{max}$ with agreeable processing times and deadlines (reduction to the Partition problem). On the other hand, Liu et al. (2003) establish the pseudo-polynomial solvability of the unbounded batch machine scheduling problem with job release dates and any regular objective.

Non-Identical Job Sizes The problems $1 \mid p - batch, B, s_j \mid \sum T_j$ and $1 \mid p - batch, B, s_j \mid \sum w_j T_j$ are strongly $\mathcal{NP}$-hard. Starting from the strongly $\mathcal{NP}$-hard problem $1 \mid p - batch, B, s_j \mid C_{max}$ (Uzsoy, 1994), it is deduced by applying the reduction rules that $1 \mid p - batch, B, s_j \mid C_{max} \propto 1 \mid p - batch, B, s_j \mid L_{max} \propto 1 \mid p - batch, B, s_j \mid \sum T_j \propto 1 \mid p - batch, B, s_j \mid \sum w_j T_j$. Given that non-identical job sizes do not make the problem easier, the strongly $\mathcal{NP}$-hard problem $1 \mid p - batch, b < n \mid L_{max}$ can also be reduced to $1 \mid p - batch, B, s_j \mid \sum T_j$ and $1 \mid p - batch, B, s_j \mid \sum w_j T_j$.

Incompatible Job Families Mehta and Uzsoy (1998) show that $1 \mid p\text{–}batch, b < n, fmls \mid \sum T_j$ is $\mathcal{NP}$-hard in the strong sense, in particular if the number of families and the batch machine capacity are arbitrary. But, they propose a DP algorithm with polynomial run time when the number of job families and the batch machine capacity are fixed. Based on the reduction rules, it follows that $1 \mid p - batch, b < n, fmls \mid \sum T_j \propto 1 \mid p - batch, b < n, fmls \mid \sum w_j T_j$.

Parallel Machines The problem $P \mid p - batch, b < n \mid \sum T_j$ is strongly $\mathcal{NP}$-hard by reduction to the strong $\mathcal{NP}$-hard problem $P \mid\mid C_{max}$ (Garey and Johnson, 1978) via $P \mid\mid \sum L_{max} \propto P \mid p - batch, b < n \mid \sum L_{max}$. Obviously $P \mid p - batch, b < n \mid \sum T_j \propto P \mid p - batch, b < n \mid \sum w_j T_j$.

6.2.5 Unit Penalties

Minimizing $\sum U_j$ on a single unbounded BPM is polynomially solvable whereas minimizing the weighted case $\sum w_j U_j$ is binary $\mathcal{NP}$-hard (Brucker et al., 1998); cf. (Baptiste, 2000; Brucker, 2007) Brucker et al. (1998) proof binary $\mathcal{NP}$-hardness for $1 \mid p-batch \mid \sum w_j U_j$ by a reduction from the binary $\mathcal{NP}$-complete problem Partition. In fact $1 \mid p-batch \mid \sum w_j U_j$ is said to be $\mathcal{NP}$-hard in the ordinary sense since Baptiste et al. (2004) show that a pseudo-polynomial algorithm exists, which means that it is not strongly $\mathcal{NP}$-hard (unless $\mathcal{P}$=$\mathcal{NP}$); cf. (Brucker, 2007).

It is also interesting to know that the corresponding unit capacity machine problem $1 \mid\mid \sum w_j U_j$ is proved to be $\mathcal{NP}$-hard with respect to binary encoding and can be solved pseudo-polynomially (Lawler, 1977). The proof is based on a reduction to the $\mathcal{NP}$-hard Knapsack problem that is equivalent to the special case of $1 \mid\mid \sum w_j U_j$ with all due dates being equal (Pinedo, 2008).

The bounded case $1 \mid p-batch, b < n \mid \sum U_j$ is strongly $\mathcal{NP}$-hard by reduction to $1 \mid p-batch, b < n \mid L_{max}$ for which Brucker et al. (1998) present the complexity proof based upon a reduction from the unary $\mathcal{NP}$-complete problem 3-Partition; cf. (Brucker, 2007). However, special cases can be solved polynomially. Lee et al. (1992) give algorithms with polynomial run time for special cases of $1 \mid p-batch, b < n \mid \sum U_j$, i.e. for the case with identical processing times and for the case with agreeable processing times and due dates.

The problem $1 \mid p-batch, b < n \mid \sum w_j U_j$ is unary $\mathcal{NP}$-hard since $1 \mid p-batch, b < n \mid \sum U_j \propto 1 \mid p-batch, b < n \mid \sum w_j U_j$. The problem becomes easier when assuming some restrictions. In the case of agreeable processing times and due dates the problem loses strong $\mathcal{NP}$-hardness; The problem $1 \mid p-batch, b < n, agreeable(p_j, d_j) \mid \sum w_j U_j$ is ordinary $\mathcal{NP}$-hard since Liu (2007) propose a pseudo-polynomial time algorithm for it.

$\mathcal{NP}$-hardness can be still maintained with identical due dates. Hochbaum and Landy (1994) give a proof for $1 \mid p-batch, b < n, d_j = d \mid \sum w_j U_j$ by a reduction from the Knapsack problem as well as pseudo-polynomial DP approach for it, i.e. $1 \mid p-batch, b < n, d_j = d \mid \sum w_j U_j$ is $\mathcal{NP}$-hard in the ordinary sense; cf. (Brucker and Kovalyov, 1996; Liu, 2007)

Polynomial solvability is achieved with identical processing times or with identical weights; $1 \mid p-batch, b < n, p_j = p \mid \sum w_j U_j$ can be solved in polynimal time (Hochbaum and Landy, 1994; Brucker, 2007) as well as $1 \mid p-batch, b < n, w_j = w \mid \sum w_j U_j$ (Hochbaum and Landy, 1994; Brucker and Kovalyov, 1996).

Release Dates The problem $1 \mid p-batch, b < n, r_j \mid \sum U_j$ is unary $\mathcal{NP}$-hard, since Lenstra et al. (1977) proof strong $\mathcal{NP}$-hardness for the unit capacity problem $1 \mid r_j \mid L_{max}$; cf. (Pinedo, 2008). A chain of reductions may be established with $1 \mid r_j \mid L_{max} \propto 1 \mid r_j \mid \sum U_j \propto 1 \mid p-batch, b < n, r_j \mid \sum U_j \propto 1 \mid p-batch, b < n, r_j \mid \sum w_j U_j$. The problem $1 \mid p-batch, b < n, r_j \mid \sum U_j$ still remains strongly $\mathcal{NP}$-hard with agreeable release dates and due dates, but becomes polynomial time solvable when the processing times are also agreeable with the release times and due dates of the jobs (Li and Lee, 1997). The problem $1 \mid p-batch, b < n, r_j \mid \sum w_j U_j$ with identical processing times is also solvable in polynomial time (Baptiste, 2000); cf. (Lee et al., 1992; Brucker, 2007).

The unbounded problem is pseudo-plynomially solvabel since Liu et al. (2003) establish pseudo-polynomial solvability for $1 \mid p-batch, r_j \mid \cdot$ with any regular objective; cf. Liu (2007).

Non-Identical Job Sizes $1 \mid p-batch, B, s_j \mid \sum U_j$ is strongly $\mathcal{NP}$-hard. Uzsoy (1994) show that $1 \mid p-batch, B, s_j \mid C_{max}$ is strongly $\mathcal{NP}$-hard, since the problem with identical processing times is equivalent to the bin-packing problem that is strongly $\mathcal{NP}$-hard. A chain of reduction may be have the form $1 \mid p-batch, B, s_j \mid C_{max} \propto 1 \mid p-batch, B, s_j \mid L_{max} \propto 1 \mid p-batch, B, s_j \mid \sum U_j \propto 1 \mid p-batch, B, s_j \mid \sum w_j U_j$. Given that the existence of non-identical jobs sizes definitely not makes the problem easier, another chain arises when starting from the L_{max} result: $1 \mid p-batch, b < n \mid L_{max} \propto 1 \mid p-batch, b < n \mid \sum U_j \propto 1 \mid p-batch, B, s_j \mid \sum U_j \propto 1 \mid p-batch, B, s_j \mid \sum w_j U_j$.

Incompatible Job Families The problem $1 \mid p-batch, b < n, fmls \mid \sum U_j$ is strongly $\mathcal{NP}$-hard. Jolai (2005) present a proof for $\mathcal{NP}$-hardness based on the id-encoding, Liu (2007) shows binary $\mathcal{NP}$-hardness, and Liu and Zhang (2008) finally establish strong/unary $\mathcal{NP}$-hardness for it. Thus the weighted case $\sum w_j U_j$ also belongs to the class of unary $\mathcal{NP}$-hard problems since $1 \mid p-batch, b < n, fmls \mid \sum U_j \propto 1 \mid p-batch, b < n, fmls \mid \sum w_j U_j$; cf. (Liu, 2007).

Jolai (2005) present a pseudo-polynomial algorithm for the special case of $1 \mid p-batch, b < n, fmls \mid \sum U_j$ with a common due date job for each family as well as a polynomial DP algorithm when the number of job families is fixed.

Parallel Machines The problem $P \mid p-batch, b < n \mid \sum U_j$ is strongly $\mathcal{NP}$-hard by reduction to $P \mid\mid C_{max}$ for which Garey and Johnson (1978) proof $\mathcal{NP}$-hardness; $P \mid\mid C_{max} \propto P \mid p-batch, b < n \mid C_{max} \propto P \mid p-batch, b < n \mid L_{max} \propto P \mid p-batch, b < n \mid \sum U_j \propto P \mid p-batch, b < n \mid \sum w_j U_j$. Another chain of reductions exists when starting from the strong $\mathcal{NP}$-hardness proof for $1 \mid p-batch, b < n \mid L_{max}$ (Brucker et al., 1998): $1 \mid p-batch, b < n \mid L_{max} \propto P \mid p-batch, b < n \mid L_{max} \propto P \mid p-batch, b < n \mid \sum U_j \propto P \mid p-batch, b < n \mid \sum w_j U_j$.

Lin and Jeng (2004) present a pseudo-polynomial DP for $P \mid p-batch, b < n \mid \sum U_j$ when the number of machines is fixed.

6.2.6 Cycle Time

The complexity of the problems of minimizing $\sum C_j$ and $\sum w_j C_j$ on a single BPM remains open (Brucker et al., 1998); cf. (Liu, 2007; Brucker and Knust, 2014). However, there exist pseudo-polynomial algorithms for special cases of $1 \mid p-batch, b < n \mid \sum C_j$ (Hochbaum and Landy, 1997; Brucker et al., 1998; Poon and Yu, 2004). In the case of identical processing times even the weighted case $1 \mid p-batch, b < n, p_j = p \mid \sum w_j C_j$ becomes polynomially solvable since a full batch schedule leads to the optimum when jobs are sequenced in order of largest to smallest weight (Albers and Brucker, 1993; Webster and Baker, 1995); cf. (Brucker, 2007). Furthermore, the unbounded problem $1 \mid p-batch \mid \sum w_j C_j$ is in $\mathcal{P}$ (Brucker et al., 1998), too.

Release Dates The existence of release dates makes the problem harder to solve. The problem $1 \mid p-batch, b < n, r_j \mid \sum C_j$ is unary $\mathcal{NP}$-hard even if $b = 1$ (Brucker et al., 1998), i.e. $1 \mid r_j \mid \sum C_j$ is unary $\mathcal{NP}$-hard (Lenstra et al., 1977).

The problem looses $\mathcal{NP}$-hardness when the processing times are equal. Webster and Baker (1995) present a DP algorithm with polynomial run time for $1 \mid p-batch, b < n, r_j \mid \sum C_j$ and Baptiste (2000) give polynomial DP a algorithm for $1 \mid p-batch, b < n, r_j \mid \sum w_j C_j$; cf. (Brucker, 2007).

It is also interesting to know that even the unbounded cases remain $\mathcal{NP}$-hard. The problem $1 \mid p-batch, r_j \mid \sum C_j$ is $\mathcal{NP}$-hard with respect to id-encoding (Liu, 2007; Liu et al., 2010). The problem $1 \mid p-batch, r_j \mid \sum w_j C_j$ is ordinary $\mathcal{NP}$-hard in the ordinary sense (Liu, 2007), i.e. binary $\mathcal{NP}$-hard by reduction to the Partition problem (Deng et al., 2004) and pseudo-polynomially solvable since Liu et al. (2003) establish pseudo-polynomial solvability of the unbounded batch machine scheduling problem with job release dates and any regular objective.

Non-Identical Job Sizes Similar to release dates, the existence of non-identical job sizes makes the problem harder to solve. Uzsoy (1994) show that $1 \mid p-batch, B, s_j \mid \sum C_j$ is strongly $\mathcal{NP}$-hard, even in the case of identical processing times. Thus the weighted case is also strongly $\mathcal{NP}$-hard since $1 \mid p-batch, B, s_j \mid \sum C_j \propto 1 \mid p-batch, B, s_j \mid \sum w_j C_j$. It is important to note that this complexity result is valid for identical processing times, which makes it also applicable for the problems based on the family processing time model, i.e. $1 \mid p-batch, B, s_j, fmls \mid \sum C_j$.

Incompatible Job Families Uzsoy (1995) show that $1 \mid p-batch, b < n, fmls \mid \sum w_j C_j$ is polynomially solvable; cf. (Liu, 2007). Obviously, the same procedure is applicable to the unweighted case; $1 \mid p-batch, b < n, fmls \mid \sum C_j \propto 1 \mid p-batch, b < n, fmls \mid \sum w_j C_j$ However, Chandru et al. (1993b) give a DP algorithm for the problem $1 \mid p-batch, b < n, fmls \mid \sum C_j$ with polynomial run time when the number of job families is fixed.

Parallel Machines As to the single machine problem, the complexity status of $P \mid p-batch, b < n \mid \sum C_j$ remains open. In contrast, the weighted variant $P \mid p-batch, b < n \mid \sum w_j C_j$ is unary $\mathcal{NP}$-hard by reduction to the strongly $\mathcal{NP}$-hard unit capacity problem $P \mid\mid \sum w_j C_j$ (T'kindt and Billaut, 2006); cf. (Brucker and Knust, 2014).

6.3 Exact Methods

The most obvious idea to solve a COP is to just enumerate all feasible solutions. For example, Wang and Chou (2013) describe an exhaustive enumeration in order to solve a single BPM scheduling problem with a bi-criterion objective, i.e. $1 \mid p-batch, b<n, B, s_j, r_j \mid C_{max}, \sum w_j T_j$. But due to the complexity of most COPs, a simple complete enumeration will result in unacceptable high computing times. The challenge is to develop efficient algorithms that perform better than a simple enumeration (Lee, 2004).

The class of exact methods to solve COPs covers MIP, DP, CP, and B&B. Commercial software packages for mathematical programming, e.g. CPLEX, Gurobi, and Xpress have seen tremendous progress over the last decade in terms of capabilities for solving much larger problem sizes (Méndez et al., 2006).

However, for a considerable amount of optimization problems present in academia and industry, it is intractable to obtain optimal solutions by the use of any exact method in a reasonable time. The crucial point is that exact methods need large amounts of time to optimally solve $\mathcal{NP}$-hard problems of practical size. Consequently, the use of exact methods becomes inapplicable for most systems in practice, where a responsible person has to make a decision as soon as possible in order to achieve desirable results (Marti and Reinelt, 2011). See the table in Appendix C for an overview of run times of exact methods for common batch scheduling problems.

Many authors share the opinion that exact methods do not seem to be the method of choice in real-world scheduling systems. Talbi (2009) notes that partial enumerative algorithms such as B&B are limited to rather small instances and thus are not advisable to solve medium and large instances. Potts and Strusevich (2009) notice a stagnation of research on B&B algorithms (and other enumerative approaches), identifying the combinatorial growth of the solution space as an obstacle to the exact solution of practical problems. Although Mönch et al. (2011a) consider MIP and CP as important solution techniques, they argue that these techniques are assessed to be too slow to be adopted in real-world implementations. Dorigo and Stützle (2004), among others, propose approximate methods that trade optimality for efficiency, since the performance of exact algorithms is not satisfactory and their applicability is often limited to rather small instances.

Klemmt et al. (2011) show that MIP approaches are basically suitable for optimizing batch small or medium scheduling problems, but only combined with decomposition methods. In awareness of the observation that even state-of-the-art MIP solver can only handle relatively small problems, Klemmt et al. (2009) propose decomposition approaches in order to reduce the problem complexity. The commercial solver CPLEX is most often used as state-of-the-art solver for MIP formulations. However, even the finding of a feasible solution can be a problem for bigger problem instances (Klemmt et al., 2008).

6.3.1 Dynamic Programming (DP)

DP is a mathematical technique providing a systematic procedure for determining the optimal sequence of decisions. It transforms a problem into a sequence of interrelated sub problems arranged in stages, employing problem-specific knowledge to a certain degree. DP is based on the principle of optimality for dynamic programming, meaning that an optimal decision in a certain stage depends on that state only and is independent of the policy decisions adopted in previous stages (Hillier and Lieberman, 2001); cf. (Ventura, 2008). The strategy is to avoid a total enumeration of the search space by pruning partial decision sequences that cannot lead to the optimal solution (Talbi, 2009). In other words, DP systematically explores the solution space without unnecessary repetitions (Reingold, 2010). Given that the problem instance is not too large, DP is known to solve $\mathcal{NP}$-hard problems (e.g. Knapsack) in pseudo-polynomial time (Brucker, 2007).

To our best knowledge, DP approaches exist only for single BPMs scheduling problems, but not for parallel BPMs. There exist DP algorithms developed for both the burn-in model and the diffusion/oxidation model, i.e. for the longest job processing time model (compatible job families) and for the family processing time model (incompatible job families). For both model types DP approaches that consider dynamic arrivals have been developed. The majority of DP algorithms is developed for special (simplified) cases in order to establish polynomial or pseudo-polynomial run time.

Brucker and Kovalyov (1996) propose a polynomial DP algorithm for $1 \mid p-batch, b<n \mid \sum w_jU_j$ with equal weights. Brucker et al. (1998) prove that a DP scheme solves $1 \mid p-batch, b<n \mid \sum C_j$ in polynomial time for fixed b where $b>1$. Moreover, the problem $1 \mid p-batch, b<n \mid \cdot$ with identical processing times can be solved polynomially by DP algorithms for the objectives $\sum w_jC_j$, $\sum T_j$, and $\sum w_jU_j$ (Brucker, 2007).

Special cases of the related problem with release dates ($1 \mid p-batch, b<n, r_j \mid \cdot$) are also solvable in polynomial time by DP. Sung et al. (2002) propose a DP algorithm for $1 \mid p-batch, b<n, r_j \mid C_{max}$ with polynomial run time when the number of job families is fixed. Webster and Baker (1995) and Baptiste (2000) examine the problem $1 \mid p-batch, b<n, r_j \mid \cdot$ with identical processing times, i.e. $1 \mid p-batch, b<n, r_j, p_j=p \mid \cdot$. For this problem type, there exist polynomial DP approaches leading to optimal solutions with minimum $\sum C_j$ (Webster and Baker, 1995), L_{max} (Webster and Baker, 1995; Baptiste, 2000), $\sum w_jU_j$ (Baptiste, 2000), and $\sum w_jC_j$ (Baptiste, 2000). Koehler and Khuller (2013) present a polynomial DP approach for $1 \mid p-batch, b<n, r_j \mid \cdot$ with identical processing times and deadlines, minimizing C_{max} and the number of batches at the same time.

Let us now consider the problems with incompatible job families. Liu and Zhang (2008) present a DP approach solving $1 \mid p-batch, b<n, fmls \mid \sum w_jU_j$ without any additional restrictions. When the number of families is fixed, there exist polynomial DP algorithms for solving $1 \mid p-batch, b<n, fmls \mid \sum C_j$ (Chandru et al., 1993b) and $1 \mid p-batch, b<n, fmls \mid \sum U_j$ (Jolai, 2005). Mehta and Uzsoy (1998) show that a DP approach polynomially solves $1 \mid p-batch, b<n, fmls \mid \sum T_j$ when the number of job families and the batch machine capacity are fixed.

Taking job arrivals into account, Tajan et al. (2011) develop a DP algorithm for $1 \mid p-batch, b<n, r_j, fmls \mid \sum C_j$.

6.3.2 Branch and Bound (BnB)

Partial enumeration methods, such as A* and B&B, basically employ tree search accompanied with strategies for pruning non-expedient subtrees, which do not contain any optimal solution. Facilitated by lower/upper bounds B&B methods eliminate a large number of feasible solutions, Today's general purpose IP/MIP software employs B&B methods to a large extend (Chandru and Rao, 2010); cf. (Talbi, 2009).

To our best knowledge, B&B approaches exist only for single BPMs scheduling problems, but not for parallel BPMs. Chandru et al. (1993a) develop a B&B for the problem $1 \mid p-batch, b<n \mid \sum C_j$ and Uzsoy and Yang (1997) consider the weighted case $1 \mid p-batch, b<n \mid \sum w_jC_j$. Sung and Choung (2000) include release dates, presenting a B&B for $1 \mid p-batch, b<n, r_j \mid C_{max}$

A remarkable amount of B&B schemes involves non-identical job-sizes ($1 \mid p-batch, B, s_j \mid \cdot$). Uzsoy (1994) present a B&B algorithm for $1 \mid p-batch, B, s_j \mid \sum C_j$ and Azizoglu and Webster (2000) additionally include job weights, i.e. solve $1 \mid p-batch, B, s_j \mid \sum w_jC_j$ with a B&B scheme. Both Dupont and Dhaenens-Flipo (2002) and Parsa et al. (2010) propose B&B schemes for $1 \mid p-batch, B, s_j \mid C_{max}$.

Lets turn to the family processing time model, i.e. to those models with incompatible job families. Based on the family processing model, Azizoglu and Webster (2001) consider non-identical job sizes in combination with incompatible families, proposing a B&B algorithm for $1 \mid p-batch, B, s_j, fmls \mid \sum w_jC_j$. Korkmaz (2004) and Yao et al. (2012) include job arrivals and solve $1 \mid p-batch, b<n, r_j, fmls \mid \sum C_j$ with B&B schemes, whereby Korkmaz (2004) assume constant processing times. Similarly, but for different objective, Tangudu and Kurz (2006) present a B&B method for solving $1 \mid p-batch, b<n, r_j, fmls \mid \sum w_jT_j$.

6.3.3 Mixed Integer Programming (MIP)

IP deals with mathematical models in which all of the decision variables are restricted to discrete values, respectively integer values. IP is used to solve discrete optimization problems, respectively COPs concerned with routing, scheduling, layout, and network design. These models consist of linear equations and inequalities on integer valued decision variables.

There exists a wide range of proprietary (and non-proprietary) software in terms of packages

and libraries wherein a general-purpose solver based on B&B is the driving engine, e.g. CPLEX[8], Gurobi[9], and Xpress[10].

If an IP model also allows continuous variables, both integer and continuous variables coexist, this model is referred to as MIP model (Chandru and Rao, 2010); cf. (Hillier and Lieberman, 2001; Weng, 2008). MIP models generalize LP and IP models.

MIP models exist for both single BPM and parallel BPMs scheduling problems. The majority of them deals with non-identical jobs sizes.

Melouk et al. (2004) and Damodaran et al. (2006, 2007) give MIP formulations for the problem $1 \mid p-batch, B, s_j \mid C_{max}$. Erramilli and Mason (2008) propose one for $1 \mid p-batch, B, s_j \mid \sum w_j T_j$. Chang et al. (2004), Xu and Bean (2007), Cheng et al. (2012) and Cheng et al. (2013) consider a related parallel machine case, i.e. $P \mid p-batch, B, s_j \mid C_{max}$ and Cheng et al. (2012) solve $P \mid p-batch, B, s_j \mid \sum C_j$.

Chou and Wang (2008) and Mathirajan et al. (2010) include job release dates, solving $1 \mid p-batch, B, s_j, r_j \mid \sum w_j T_j$ with MIP. Vélez-Gallego et al. (2011) and Xu et al. (2012) examine $1 \mid p-batch, B, s_j, r_j \mid C_{max}$. Chung et al. (2009), Wang and Chou (2010) and Ozturk et al. (2012) consider the related parallel machine case $P \mid p-batch, B, s_j, r_j \mid C_{max}$, whereby Ozturk et al. (2012) assume equal processing times.

Let us turn to the family scheduling model. Dauzère-Pérès and Mönch (2013) present a MIP formulation for $1 \mid p-batch, b<n, fmls \mid \sum U_j$ and the weighted variant $\sum w_j U_j$. Dobson and Nambimadom (2001) solve $1 \mid p-batch, B, s_j, fmls \mid \sum w_j C_j$ with MIP and Koh et al. (2005) give a MIP formulation for $1 \mid p-batch, B, s_j, fmls \mid \cdot$ with the objectives C_{max}, $\sum C_j$ and $\sum w_j C_j$.

Klemmt et al. (2009) develop a MIP formulation for the problem $P \mid p-batch, b<n, r_j, fmls \mid \sum w_j T_j$ with machine eligibility constraints. Klemmt et al. (2008, 2011) develop MIP models for $P \mid p-batch, B, s_j, r_j, fmls \mid \cdot$ with machine eligibility constraints and deadlines in their models, minimizing C_{max}, $\sum C_j$, $\sum w_j C_j$, or $\sum w_j T_j$. Cakici et al. (2013) provides a MIP for $P \mid p-batch, B, s_j, r_j, fmls \mid \sum w_j C_j$.

Kempf et al. (1998) and Gokhale and Mathirajan (2011, 2014) consider special cases. Gokhale and Mathirajan (2011) allows job splitting for the problem $1 \mid p-batch, B, s_j, r_j, fmls \mid \sum w_j T_j$ and present a MIP for the same problem with parallel machines in (Gokhale and Mathirajan, 2014). Kempf et al. (1998) examine the model $1 \mid p-batch, B, s_j, fmls \mid \cdot$ with secondary resources, minimizing C_{max} and $\sum C_j$.

6.3.4 Constraint Programming (CP)

CP is based on tree search and logical implications, whereas the underlying model consists of a set of variables with values out of a finite domain of integers, linked by a set of constraints. The CP paradigm basically models the properties of the desired solution. In general, declarative models in CP are more compact than MIP models, which does not mean that CP models generally perform better than MIP models. The efficiency of CP/MIP models and their solvers mainly depends on the structure of the problem and in particular on the model. In general, it is said that CP models perform better for tight constrained problems such as scheduling problems than for problems with a large number of feasible solutions (Talbi, 2009); cf. (Pinedo, 2008).

CP approaches still represent a minority in the area of scheduling. Malapert et al. (2012) describe the use of CP in order to solve the problem $1 \mid p-batch, B, s_j \mid L_{max}$. They show that CP clearly outperforms other exact approaches based on mathematical formulation or B&B.

Kanet et al. (2004) examine CP approaches for scheduling problems more in detail. It could be shown that CP methods are particularly suitable to solve scheduling problems that involve sequencing and resource constraints. However, their applicability for solving more general scheduling problems remain controversial (Méndez et al., 2006).

[8] http://www.ibm.com
[9] http://www.gurobi.com
[10] http://www.fico.com

6.4 Approximation Algorithms

An approximation algorithm for a certain problem class guarantees that any obtained solution corresponds to an objective value for which a factor p defines the distance to the actual optimum. For minimization problems, p is defined as $p = 1 + \varepsilon$ with $\varepsilon > 0$ (Brucker, 2007; Chandru and Rao, 2010; Klein and Young, 2010). An approximation algorithm that exclusively generates solutions with a worst-case ratio p, respectively leading to solutions having an objective value not worse than a factor p times the optimum, is referred to as p-approximation algorithm. An approximation scheme frames a number of p-approximation algorithms for a given problem class. If an approximation algorithm solves a problem under online setting, its performance is given with a worst-case ratio compared to the offline case.

A PTAS is an approximation scheme with a time complexity that is polynomial in the input size (Pinedo, 2008; Talbi, 2009). Since the definition of a PTAS only requires polynomial time complexity in the input size, a PTAS may have a time complexity that grows exponentially in $\frac{1}{\varepsilon}$: if ε decreases, the computation time will increase dramatically (Pinedo, 2008).

A FPTAS is a PTAS with an additional constraint that bounds its time complexity. FPTAS is an approximation scheme with a time complexity that is polynomial in the input size as well as in $\frac{1}{\varepsilon}$ (Pinedo, 2008; Talbi, 2009). With regard to worst-case approximations, an FPTAS is the strongest possible result that one can obtain for a $\mathcal{NP}$-hard problem (Pinedo, 2008).

Brucker and Kovalyov (1996) give a FPTAS for $1 \mid p-batch, b<n \mid \sum w_j U_j$. Hochbaum and Landy (1997) give an approximation algorithm for $1 \mid p-batch, b<n \mid \sum C_j$ and Cai et al. (2002) and Deng et al. (2002) give a PTAS for it. Chen et al. (2001) develop an approximation scheme for $1 \mid p-batch, b<n \mid \sum w_j C_j$ under online setting. Liu and Yu (2000) propose an algorithm with performance bound for $1 \mid p-batch, b<n, r_j \mid C_{max}$. Poon and Zhang (2000) and Deng et al. (2003) propose PTAS for the same problem. Liu and Cheng (2005) give a PTAS for $1 \mid p-batch, b<n, r_j \mid \sum C_j$. Cao and Yang (2009) and Lu et al. (2009b) examine the problem $1 \mid p-batch, b<n, r_j \mid \cdot$ with rejections. They provide PTASs for an objective function that involves C_{max} and penalty costs for rejections. Zhang et al. (2001a) study the problem $1 \mid p-batch, b<n, r_j \mid C_{max}$ under online setting, providing algorithms with performance bounds. Li et al. (2005a) present an approximation algorithm for $1 \mid p-batch, B, s_j, r_j \mid C_{max}$. Zhang et al. (2001b), Zhang and Cao (2007) and Kashan et al. (2009) give approximation algorithms for $1 \mid p-batch, B, s_j \mid C_{max}$.

Li et al. (2004) present a PTAS for $Pm \mid p-batch, b<n, r_j \mid L_{max}$. Li et al. (2005b) and Zhang et al. (2005) give PTAS for $Pm \mid p-batch, b<n, r_j \mid C_{max}$ Li et al. (2012a) give an algorithm with performance bound for the problem $Pm \mid p-batch, b<n, r_j \mid C_{max}$ under online setting, allowing their algorithm to use information from a time window look ahead. Li (2012) and Ozturk et al. (2012) present approximation algorithms for $Pm \mid p-batch, B, s_j, r_j \mid C_{max}$, whereby Ozturk et al. (2012) assumes identical processing times. Cheng et al. (2012) give polynomial approximation algorithm for $Pm \mid p-batch, B, s_j \mid C_{max}$ and $Pm \mid p-batch, B, s_j \mid \sum C_j$.

Li et al. (2012b) provide a competitive ratio for their online algorithm solving a variant of $Pm \mid p-batch, b<n, fmls \mid \cdot$ with the objective total weighted earliness. Bar-Noy et al. (2009) study the problem $Pm \mid p-batch, b<n, r_j, fmls \mid \cdot$ with deadlines, providing an approximation algorithm that maximizes the weight of the scheduled jobs.

Nong et al. (2008b) and Meng and Lu (2011) present online approximation algorithms for $1 \mid p-batch, b<n, fmls \mid C_{max}$. Although their models consider incompatible job families, the processing time of the batch equals the longest processing time of the jobs in the batch. Nong et al. (2008a) provide approximation algorithms for $1 \mid p-batch, b<n, r_j, fmls \mid C_{max}$ and $1 \mid p-batch, B, s_j, r_j, fmls \mid C_{max}$.

6.5 Heuristics

Constructive heuristics are the method of choice to create solutions for deterministic scheduling problems in a very short time. Construction algorithms describe an incremental procedure. Starting from an empty initial solution, they iteratively add solution components until a complete solution is obtained without any backtracking. At each step an additional solution component is chosen from a ranked list based on some heuristic information. Constructive heuristics are usually problem-specific

and non-iterative, they create one single solution by applying a set of rules based on problem-specific knowledge; cf. (Dorigo and Stützle, 2009; Talbi, 2009; Zäpfel et al., 2010).

Among others Leung (2004), Pinedo (2005, 2008) and T'kindt and Billaut (2006) examine various relatively simple priority or dispatching rules for solving deterministic scheduling models. Blackstone et al. (1982), Haupt (1989) and Kemppainen (2005) give surveys of dispatching rules.

Dispatching rules can be classified in various ways, depending on the type of information used for job prioritization. For example, it is distinguished between local queue information and global shop information. Local rules only use information related to the machine or tool group for decision making, whereas global rules involve additional information from other machines or even from the entire shop floor; cf. (Haupt, 1989; Dabbas and Fowler, 2003; Pinedo, 2008). Another distinction is made between time-independent (static) and time-dependent (dynamic) rules. Static rules are just a function of the job and/or of the machine data whereas dynamic rules imply that the priority of jobs may change as time proceeds cf. (Haupt, 1989; Pinedo, 2008). Dispatching rules may be further classified according to the type of attributes used to create an ordered list of jobs; (Blackstone et al., 1982; Haupt, 1989; Dabbas and Fowler, 2003). Blackstone et al. (1982) distinguish between: *a*) rules involving processing time, *b*) rules involving due dates, *c*) rules involving shop characteristics or job characteristics other than processing times or due dates, *d*) rules involving one or more of the first three categories (composite rules).

6.5.1 Basic Rule-Based Heuristics

There exist numerous rule-based algorithms for various problems, ranging from the simplest sorting schemes to quite complex heuristic algorithms. In this section, three basic dispatching rules are briefly introduced.

First In First Out (FIFO) One of the simplest rules is to process jobs in the order of their arrival. This rule is also referred to as First In First Out (FIFO) rule (Rose, 1998), Earliest Release Time First (ERT) rule (Sung and Choung, 2000), Earliest Start Time First (EST) rule (T'kindt and Billaut, 2006), Earliest Release Date First (ERD) rule, or First Come First Served (FCFS) rule (Pinedo, 2008). FIFO attempts to equalize the waiting times of the jobs.

Ikura and Gimple (1986) propose the FOE algorithm to solve batching problems. The idea of the FOE algorithm is to create as many full batches as possible, while only allowing the first batch to be partially empty. FOE leads to an optimal solution for $1 \mid p-batch, b<n, r_j, p_j=p \mid C_{max}$ when the jobs are scheduled in FIFO order; cf. (Webster and Baker, 1995; Sung and Choung, 2000).

Shortest Processing Time (SPT) Among the most traditional rules, we find simple local rules that use job information such as processing time. For example, the Shortest Processing Time (SPT) rule schedules jobs according to their processing time, i.e. the job with smallest processing time is processed first. The SPT rule leads to optimal schedules for $1 \mid\mid \sum C_j$ and $P \mid\mid \sum C_j$ (Leung, 2004; T'kindt and Billaut, 2006; Pinedo, 2008). Webster and Baker (1995) note that a full batch schedule leads to optimum for the problem $1 \mid p-batch, b<n, p_j=p \mid \sum C_j$

The Weighted Shortest Processing Time (WSPT) rule extends SPT by involving job weights, leading to a job ranking in decreasing order of $\frac{w_j}{p_j}$. The WSPT rule is optimal for $1 \mid\mid \sum w_j C_j$ (Leung, 2004; Pinedo, 2008); cf. (T'kindt and Billaut, 2006). Webster and Baker (1995) note that a full batch schedule leads to the optimum for the problem $1 \mid p-batch, b<n, p_j=p \mid \sum C_j$ when the jobs are sequenced in order of the largest to the smallest weight.

Earliest Due Date (EDD) There exist several rules involving due date information, e.g. Critical Ratio (CR), Minimum Slack (MS) and Earliest Due Date (EDD) among many others. EDD, as one of the most popular rules, sequences the jobs in increasing order of their due date d_j. A schedule created with the EDD rule is optimal for the problem $1 \mid\mid \sum L_{max}$ (T'kindt and Billaut, 2006); cf. (Leung, 2004; Pinedo, 2008).

Webster and Baker (1995) note that a full batch schedule leads to optimum for the problem $1 \mid p-batch, b<n, p_j=p \mid \sum L_{max}$ when the jobs are sequenced in order of the smallest to the largest due date, i.e. by the EDD rule. The Weighted Earliest Due Date (WEDD) rule extends

EDD by involving job weights, leading to a job ranking in decreasing order of $\frac{w_j}{d_j}$. Kemppainen (2005) presents a ranking of dispatching rules according to the weighted mean tardiness objective.

6.5.2 Batch Apparent Tardiness Cost (BATC)

The Batch Apparent Tardiness Cost (BATC) rule plays a prominent role in this work as it is one of the most known heuristic rules to solve batching problems with due dates.

Vepsalainen and Morton (1987) develop the Apparent Tardiness Cost (ATC) rule for minimizing the weighted tardiness in job shops. The ATC rule is a composite dispatch rule that basically combines the WSPT rule and the MS rule. The performance of ATC depends on the value of the look-ahead parameter k (Valente, 2007). Valente (2007) discusses how to improve the performance of the ATC dispatch rule by determining the look-ahead parameter value with the help of workload data.

Several authors extend the ATC rule in order to solve batch scheduling problems, at what the proposed rules are called BATC rules. Mehta and Uzsoy (1998) probably present the first variant of BATC in order to solve the problem $1 \mid p-batch, b<n, fmls \mid \sum T_j$. Balasubramanian et al. (2004) describe a BATC rule that considers weights, intended to solve the problem $P \mid p-batch, b<n, fmls \mid \sum w_j T_j$.

Habenicht and Mönch (2003) and Mönch et al. (2006b) focus on the problem $P \mid p-batch, b<n, r_j, fmls \mid \sum w_j T_j$, developing extensions of ATC that incorporate job arrivals and weights. Habenicht and Mönch (2003) investigate the performance of certain modifications of the ATC dispatching rule and extend this approach by considering future lot arrivals. Mönch et al. (2006b) propose to use inductive decision trees and neural networks for estimating the look-ahead parameter k due to the lack of a closed formula. Klemmt et al. (2011) discuss different variants of the BATC rule for the problem $R \mid p-batch, b<n, B, sj, r_j, fmls \mid \cdot$ with machine eligibility constraints and deadlines, minimizing C_{max}, $\sum w_j T_j$ or $\sum w_j C_j$.

6.5.3 More Problem-Specific Batching Heuristics

This section provides an overview on problem-specific batching heuristics beyond the basic dispatching and priority-based rules discussed in the previous section. It follows the same problem-based grouping used in the literature review and lists numerous heuristic approaches for 16 batch scheduling problem types.

$1 \mid$ p-batch, b $<$ n $\mid \cdot$ Chandru et al. (1993a) give a heuristic for minimizing $\sum C_j$ and Uzsoy and Yang (1997) give consider the weighted case $\sum w_j C_j$.

$1 \mid$ p-batch, b $<$ n, $r_j \mid \cdot$ Lee and Uzsoy (1999) and Sung and Choung (2000) present heuristics for minimizing $\mid C_{max}$.

$1 \mid$ p-batch, B, s_j, $r_j \mid \cdot$ Chang and Wang (2004) propose a heuristic for $\mid \sum C_j$ minimization. Chou and Wang (2008) and Mathirajan et al. (2010) provide heuristics in order to minimize $\sum w_j T_j$. Vélez-Gallego et al. (2011), Xu et al. (2012) and Zhou et al. (2013) present heuristics for a minimum C_{max}.

$1 \mid$ p-batch, b $<$ n, B, $s_j \mid \cdot$ Uzsoy (1994) and Parsa et al. (2010) present heuristic solutions for the objective C_{max} and Uzsoy (1994) for $\sum C_j$. Jolai Ghazvini and Dupont (1998) consider the same problem, but with the objective function that minimizes mean flow times.

Pm $\mid$ p-batch, b $<$ n $\mid \cdot$ Chandru et al. (1993a) give a heuristic for the objective $\sum C_j$. Mönch and Unbehaun (2007) present three decomposition heuristics that minimizes the difference between earliness and tardiness of the jobs, while assuming that the due dates are equal.

Pm $\mid$ p-batch, b $<$ n, $r_j \mid \cdot$ To our best knowledge, no heuristic algorithms are presented for the problem $Pm \mid p-batch, b<n, r_j \mid \cdot$ in particular.

Pm | p-batch, B, s_j, r_j | · Chung et al. (2009), Chen et al. (2010) and Damodaran and Vélez-Gallego (2010) present heuristics for minimizing C_{max}.

Pm | p-batch, b < n, B, s_j | · Damodaran and Chang (2008) and Li et al. (2013) present several heuristics for the objective C_{max}. Shao et al. (2008a,b) propose to use a neural net and Chen et al. (2011) present a clustering algorithm for the same problem.

Pm | p-batch, b < n, fmls | · Uzsoy (1995) presents heuristics for the problem with the objectives C_{max}, L_{max} and $\sum w_j C_j$ and Balasubramanian et al. (2004) develop a BATC-variant for minimizing $\sum w_j T_j$.

Pm | p-batch, b < n, r_j, fmls | · Habenicht and Mönch (2003) and Mönch et al. (2006b) examine variants of BATC for minimizing $\sum w_j T_j$. Kim et al. (2010) give a heuristic for the unweighted case $\sum T_j$.

Pm | p-batch, B, s_j, r_j, fmls | · Klemmt et al. (2011) discuss different variants of the BATC rule for the problem $R \mid p-batch, b < n, B, sj, r_j, fmls \mid \cdot$ with machine eligibility constraints and deadlines, minimizing C_{max}, $\sum w_j T_j$ or $\sum w_j C_j$. Mathirajan and Sivakumar (2006a) and Gokhale and Mathirajan (2014) describe a few greedy heuristics for $\sum w_j T_j$, whereby Gokhale and Mathirajan (2014) allows job splitting.

Pm | p-batch, B, s_j, fmls | · Koh et al. (2004) examine heuristics for the objectives C_{max}, $\sum C_j$ and $\sum w_j C_j$ and Payman and Leachman (2010) discuss a heuristic solution for the problem with secondary resources and a multi-objective function.

1 | p-batch, b < n, fmls | · Mehta and Uzsoy (1998) probably present the first variant of BATC in order to minimize $\sum T_j$. Devpura et al. (2001) and Perez et al. (2005) give a heuristic for the weighted case $\sum w_j T_j$. Boudhar (2003) examine a heuristic solution for the objective C_{max} with compatibility graph constraints. Sabouni and Jolai (2010) discuss a special case of $1 \mid p-batch, b < n, fmls \mid \cdot$ where the objective is minimize L_{max} and C_{max} at the same time.

1 | p-batch, b < n, r_j, fmls | · Uzsoy (1995) presents heuristics for the L_{max} objective. Boudhar (2003) examines a heuristic solution for the objective C_{max} with compatibility graph constraints. Korkmaz (2004) minimize $\sum C_j$. Kurz and Mason (2008) examine $\sum w_j T_j$. Jia et al. (2013) consider reentrant jobs and present job-family-oriented algorithm based on a rolling horizon control strategy that minimizes $\sum w_j C_j$, $\sum w_j T_j$ and L_{max}.

1 | p-batch, B, s_j, r_j, fmls | · Gokhale and Mathirajan (2011) allows job splitting for the problem $1 \mid p-batch, B, s_j, r_j, fmls \mid \sum w_j T_j$ and present heuristics for it.

1 | p-batch, B, s_j, fmls | · Hoitomt and Luh (1992) minimize $\sum T_j$ with heuristics. Dobson and Nambimadom (2001) minimize mean weighted flow time for the problem the same problem. Koh et al. (2005) study heuristics for the objectives C_{max}, $\sum C_j$ and $\sum w_j C_j$. Kempf et al. (1998) develop heuristics for the problem with secondary resources, minimizing C_{max} and $\sum C_j$.

6.6 Metaheuristics

Metaheuristics, and LS methods in particular, are often the methods of choice for solving real-life scheduling problems with a variety of complicating constraints, since these algorithms can obtain good quality solutions within a reasonable time (Méndez et al., 2006; Potts and Strusevich, 2009). Several authors document the superiority of metaheuristics compared to MIP based approaches in experimental studies, in particular for batch scheduling problems. From the NFLTs it can be informally deduced that no metaheuristic performs better than another across all possible problems (Wolpert and Macready, 1997). Ho and Pepyne (2001) point out that specializing search algorithms to the landscape structure of the focused problem class is the only way one strategy can outperform

another (refer to Section 5.3). Thus, it can be said that the choice of a particular metaheuristic is less important than its actual implementation.

However, some authors compare metaheuristics with each other and report results which indicate that one metaheuristic, respectively its implementation, outperforms another for a given set of problem instances. The majority of metaheuristic implementations developed to solve batch scheduling problems follows the paradigms of ACO, SA and GA. In comparison, VNS, PSO, GRASP and other metaheuristics represent a minority in this area.

6.6.1 Ant Colony Optimization (ACO)

Neto and Filho (2012) provide a review regarding ACO applied to scheduling problems in general, providing guidelines for implementation and directions for future research.

A considerable amount of works focuses on problems with the makespan objective. Jia and Leung (2014) and Cheng et al. (2010) develop a metaheuristic algorithm based on ACO for the problem $1 \mid p-batch, B, s_j \mid \sum C_{max}$. Cheng et al. (2010) consider fuzzy processing times and demonstrate that the ACO algorithm outperforms GA and SA in all instances. Xu et al. (2008a,b) describe an ACO algorithm for the same problem with $\sum w_j C_j$ objective. Cheng et al. (2013) present an ACO method for parallel machines, in particular solving $P \mid p-batch, B, s_j \mid \sum C_{max}$. Xu et al. (2012) and Zhou et al. (2013) study ACO for the problem $1 \mid p-batch, B, s_j, r_j \mid \sum C_{max}$. Xu et al. (2012) show that ACO is more robust and consistently outperforms MIP, especially for large job instances. Their results also show that the ACO algorithm outperforms an implementation of a GA, especially for large job instances. Zhou et al. (2013) also compare their ACO with a GA. Chen et al. (2010) compare ACO with GA for the problem $P \mid p-batch, B, s_j, r_j \mid \sum C_{max}$. Their results show that the GA is able to obtain better solutions when dealing with small-job instances compared to ACO, whereas ACO dominates GA in large-job instances.

Besides the works that focus on makespan, there are works presenting ACO algorithms for other objectives, e.g for OTD. Xu et al. (2013) develop an ACO method for the problem $P \mid p-batch, B, s_j, r_j \mid \cdot$ in order to minimize $\sum C_{max}$ and $\sum L_{max}$ at the same time. Kashan and Karimi (2007) analyze the performance of ACO for $1 \mid p-batch, B, s_j \mid \sum w_j C_j$. Almeder and Mönch (2011) compare ACO, VNS and a GA with each other for instances of the problem $Pm \mid p-batch, b<n, fmls \mid \sum w_j T_j$. Mönch and Almeder (2009) and Raghavan and Venkataramana (2006) discuss ACO approaches for the same problem. Mönch and Almeder (2009) note that that the ACO approach slightly outperforms the GA with respect to solution quality, while requiring considerably less computational time than GA. Guo et al. (2010) and Li and Qiao (2008) propose a ACO method for the problem $Pm \mid p-batch, b<n, r_j, fmls \mid \sum w_j T_j$ with sequence dependent setup times The same model is then extended with machine eligibility constraints in (Li et al., 2008), (Li et al., 2009a) and (Li et al., 2009b), where Li et al. (2009b) minimize $\sum w_j T_j$ and $C_{\max}$ at the same time.

6.6.2 Simulated Annealing (SA)

Melouk et al. (2004), Mathirajan et al. (2004) and Damodaran et al. (2007) develop SA algorithms for the problem $1 \mid p-batch, B, s_j \mid C_{max}$. In particular, Melouk et al. (2004) and Damodaran et al. (2007) state that SA outperforms CPLEX in their experiments. Erramilli and Mason (2008) describe a SA scheme for $1 \mid p-batch, B, s_j \mid \sum w_j T_j$. Chang et al. (2004) and Kashan et al. (2008) study SA algorithms for $P \mid p-batch, B, s_j \mid C_{max}$. Chang et al. (2004) note that SA outperforms CPLEX on most of the instances and Kashan et al. (2008) compare SA with a hybrid GA. Mathirajan et al. (2010) solve the problem $1 \mid p-batch, B, s_j, r_j \mid \sum w_j T_j$ with SA and claim that the SA algorithm consistently finds a robust solution in a reasonable amount of computation time. Damodaran and Vélez-Gallego (2012) compare SA with GRASP (Damodaran et al., 2011) for the problem $1 \mid p-batch, B, s_j, r_j \mid C_{max}$. Their computational experiments show that the SA approach is comparable to GRASP with respect to solution quality, and less computationally costly. Wang and Chou (2013) propose a SA scheme that solves $1 \mid p-batch, B, s_j, r_j \mid \sum w_j T_j$. Wang and Chou (2010) compare SA, GA and MIP with each other for the problem $P \mid p-batch, B, s_j, r_j \mid C_{max}$. Yugma et al. (2008) present a solution scheme based on SA in order to solve a variant of $P \mid p-batch, B, s_j, r_j \mid \cdot$ with deadlines and machine eligibility constraints in order to minimize a multi-objective function.

6.6.3 Genetic Algorithms (GAs)

Mönch et al. (2006a) study the problem $1 \mid p-batch, b<n \mid \cdot$ with deadlines and equal due dates, providing a GA that minimizes earliness and tardiness. Wang and Uzsoy (2002) develop a GA combined with DP for the problem $1 \mid p-batch, b<n, r_j \mid L_{max}$. Damodaran et al. (2006), Kashan et al. (2006a,b) study the performance of GA for the problem $1 \mid p-batch, B, s_j \mid C_{max}$. Kashan et al. (2006b) claim that the GA significantly outperforms the SA in terms of both quality of solutions and required run times. The results in (Damodaran et al., 2006) indicate that the GA is able to arrive at better makepan with shorter run times compared to a SA approach and a commercial solver. Kashan et al. (2010) develop a GA for the problem $1 \mid p-batch, B, s_j \mid \cdot$ with a bi-criterion objective involving C_{max} and L_{max}. Koh et al. (2005) develop a GA for the problem $1 \mid p-batch, B, s_j, fmls \mid \cdot$ minimizing C_{max}, $\sum C_j$ or $\sum w_j C_j$. Koh et al. (2004) consider the same objectives for the related parallel machine problem $P \mid p-batch, B, s_j, fmls \mid \cdot$. Chou et al. (2006) and Zhou et al. (2013)present GAs for the problem $1 \mid p-batch, B, s_j, r_j \mid C_{max}$, whereby Zhou et al. (2013) compare the performance of the GA with ACO among other heuristics. Chou and Wang (2008) develop a GA combined with DP for the problem $1 \mid p-batch, B, s_j, r_j \mid \sum w_j T_j$, claiming that their implementation outperforms MIP for large-job problems. Dauzère-Pérès and Mönch (2013) describe a GA for the problem $1 \mid p-batch, b<n, fmls \mid \cdot$ with the objectives $\sum T_j$ and $\sum w_j T_j$.

Besides single machine problems, there are works that propose GAs for solving problems with parallel machines. Xu and Bean (2007) and Kashan et al. (2008) develop GAs for the problem $P \mid p-batch, B, s_j \mid C_{max}$ where Kashan et al. (2008) compare their implementation with a variant of SA. Xu and Bean (2007) claim that their implementation of the GA outperforms CPLEX in terms of solutions and computation times, especially for larger problems. Chen et al. (2010) and Wang and Chou (2010) develop GA for the problem $P \mid p-batch, B, s_j, r_j \mid C_{max}$. Chen et al. (2010) show that the GA is able to obtain better solutions when dealing with small-job instances compared to ACO, whereas ACO dominates GA in large-job instances and Wang and Chou (2010) compare GAs with MIP and SA. Balasubramanian et al. (2004) and Almeder and Mönch (2011) study the performance of GAs for problem $Pm \mid p-batch, b<n, fmls \mid \sum w_j T_j$, where Almeder and Mönch (2011)compare GAs with ACO and VNS. Malve and Uzsoy (2007) present a GA for the problem $P \mid p-batch, b<n, r_j, fmls \mid L_{max}$. Habenicht and Mönch (2003) and Mönch et al. (2005) develop GAs for the problem $P \mid p-batch, b<n, r_j, fmls \mid \sum w_j T_j$. Reichelt and Mönch (2006) develop a GA for the problem $P \mid p-batch, b<n, r_j, fmls \mid \cdot$ with a multi-component objective involving C_{max} and $\sum w_j T_j$.

6.6.4 Variable Neighborhood Search (VNS)

Almeder and Mönch (2011) state that the VNS approach outperforms the ACO and the GA approach with respect to time and solution quality for the problem $Pm \mid p-batch, b<n, fmls \mid \sum w_j T_j$. Cakici et al. (2013) evaluate the performance of VNS for the problem $Pm \mid p-batch, B, s_j, r_j, fmls \mid \sum w_j C_j$. Klemmt et al. (2009) dscribe an implementation of VNS for the problem $Pm \mid p-batch, b<n, r_j, fmls \mid \sum w_j T_j$ with machine eligibility constraints. The experiments in (Klemmt et al., 2009) turn out that VNS outperforms MIP with respect to solution quality and time; they find that VNS performs faster than MIP while providing at the same time high quality solutions. Various experiments that concern the performance of VNS under varying settings for the problem $Pm \mid p-batch, b<n, r_j, fmls \mid \cdot$ are reported in (Kohn and Rose, 2012, 2013; Kohn et al., 2013).

6.7 Decomposition Methods

Pinedo (2008) describes four types of decomposition methods: *a*) machine-based decomposition, *b*) job-based decomposition, *c*) time-based decomposition, and *d*) hybrid decomposition methods that combine the three preceding decomposition types. In awareness of the observation that even state-of-the-art MIP solver can only handle relative small problems, decomposition approaches are proposed in order to reduce the problem complexity (Klemmt et al., 2009).

Machine-Based Decomposition Methods Machine-based decomposition is often used in flow shop, job shop, and open shop environments. For example, the shifting bottleneck technique

decomposes a shop scheduling problem into a number of single machine scheduling problems. The resulting single machine scheduling problems are then solved in order of their importance in a sense that the most critical machine is scheduled first (Pinedo, 2008). Fowler et al. (2003) present a modification of the shifting bottleneck heuristic for scheduling wafer fabrication facilities. In the light of parallel machine environments, it might be necessary to reduce the number of machines by dividing the set of machines into a number of subsets, preferably CMSs with disjunctive processes. It is quite easy to identify CMSs with simple iterative procedures if existent. However, in practice a particular CMS might be still too large to be solved efficiently and thus needs to be split into machine sets with overlapping processes. In this case COP arises with the task to find a number of subordinated machine groups so that the number of process overlaps is minimized. This problem is related to the cell formation problem in manufacturing. The cell formation problem attempts to group machines and part families in dedicated manufacturing cells such that the number of voids and exceptional elements in cells are minimized (Paydar and Saidi-Mehrabad, 2013); cf. (Elbenani et al., 2012; Modrak and Semanco, 2012).

Job-Based Decomposition Methods Generally a job-based decomposition method refers to the idea of creating sub problems that correspond to a reduced number of jobs or operations. By iteratively inserting jobs into a partial schedule, one sub problem after another is solved (Pinedo, 2008). For the parallel machine scheduling problem, job-based decomposition methods mean to define a number of consecutive groups of an ordered job list. As a result each sub problem refers to a group of jobs. The procedure begins to solve the scheduling problem with the first group comprising the top-priority jobs and proceeds with the remaining job-groups. cf. (Klemmt et al., 2009)

Time-Based Decomposition Methods Time-based decomposition methods are also known as rolling horizon or time-window procedures (Pinedo, 2008). Ovacik and Uzsoy (1995) originally present a family of rolling horizon heuristics for a parallel machine scheduling problem. Time-based decomposition methods create sub problems along the time axis. A partial schedule is generated up to a given point in time, ignoring everything that could happen afterwards. The focused time window is then shifted to the next time frame at which the next sub problem is solved. For most problems, this strategy indirectly reduces the number of jobs and thus reduces the problem complexity (Pinedo, 2008); cf. Klemmt et al. (2009)

Hybrid Decomposition Methods Klemmt et al. (2009) combine machine-based, time-based and job-based decomposition procedures into a hybrid approach. Based on a time-window decomposition scheme, machine-based decomposition is used in a sense that only machines (in a time window) with availability times lower than a limited time horizon are regarded. Additionally, the number of jobs to be scheduled in each time window is reduced by selecting a limited number of jobs to be scheduled from an ordered priority list. Jobs that are not regarded in a time window are then simply delayed to the following time-window.

6.8 Real-Time Control Strategies

Real-time control strategies, such as simple rule-based dispatching approaches, are typically experimentally examined with stochastic scheduling models. The stochastic scheduling model defines a finite number of jobs that have to be scheduled. The job properties, such as processing times, release dates and due dates are a priori not exactly known. These are random variables for which only their distributions are known in advance. The actual processing times become known only at the completion of the processing. Likewise, the release dates and due dates become known only at the actual occurrence of the release or due date. The decision-making system follows a control policy in order to minimize a given objective function in expectation, knowing only the probability distributions of the data in advance. Stochastic scheduling models provide a framework for evaluating relatively simple priority or dispatching rules in the context of real-time control (Pinedo, 2004, 2005, 2008). This type of scheduling model is also referred to as infinite horizon model, emphasizing the fact that this model type is usually examined with a large number of jobs in the long run (Neale and Duenyas, 2000; Tajan et al., 2012).

The stochastic scheduling model is closely related to research in Queuing Theory since emphasis is put on probability distributions of important variables, such as job release dates. The jobs stochastically enter the system over time and the performance of the studied control policy is then examined over a sufficiently long period of time. Studying a system under the assumption of continuous arrivals to a practically infinite horizon, i.e. over a long period of time, thus experimentally mirrors the system behavior studied by the Queuing Theory of manufacturing systems.

Since real-time control requires ad-hoc decisions to instantly trigger activities on the shop floor, long computation times to make decisions cannot be accepted in view of an effective operational control policy. The predominant control system in waferfabs is still the rule-based dispatching concept providing reliable real-time capability through simple heuristics (e.g. priority rules) that instantly lead to immediate decisions in a few seconds. The software system of choice is most often the APF RTD product[11] (Mönch et al., 2011a); cf. (Leachman, 2002)

Akcali et al. (2000) show that the loading policy has a more significant effect on flow time and due date performance than the dispatching policy. In other words: the decision to start or to delay a non-full batch is more important than the choice of the jobs.

Whenever a BPM completes a process and becomes available again, a scheduling decision has to be taken in order to proceed. This involves two sub-decisions: a dispatching decision and a loading decision. The dispatching decision determines which jobs have to be processed next (on which machine), referring to the prioritization of the jobs that are put together in a batch. The loading decision determines how many lots to put into the batch, considering the trade-off between starting the batch or waiting for more lots to arrive. Especially the loading decision further complicates the scheduling task at hand. On one hand, if there are less jobs available than the capacity of the machine allows, starting a partial batch immediately would lead to a waste of machine utilization. On the other hand, delaying the process start in order to wait for arriving jobs increases the queuing time for the lots that are currently waiting for processing (Akcali et al., 2000). BP is always a trade-off between machine utilization and CT of the jobs (Sha et al., 2004, 2007).

The dispatching/loading rules for controlling BPMs in real-time fall into two categories: threshold strategies and look-ahead strategies. A threshold strategy decides with knowledge about the current situation only, without any further information. Look-ahead strategies further incorporate known near-future arrivals into the decision process; cf. (Cha et al., 2012; van der Zee et al., 1997).

Reviews Mathirajan and Sivakumar (2003) review batch scheduling problems in semiconductor manufacturing, classifying batching problems into 12 groups while distinguishing between stochastic and deterministic problems. They refine their classification schemes and systematically organize the published articles in an updated survey three years later (Mathirajan and Sivakumar, 2006b). Robinson et al. (2000), van der Zee (2000, 2003) and Cha et al. (2012) particularly examine real-time control strategies for BPMs in industry, where Robinson et al. (2000) and Cha et al. (2012) focus on semiconductor manufacturing. Robinson et al. (1995), van der Zee et al. (1997, 2001), Sha et al. (2004, 2007) do not provide reviews in the classical meaning, but do provide reviews with literature related to real-time control of BPMs in their works.

6.8.1 Threshold Strategy

Neuts (1967) proposes the MBS rule, which is often used as a benchmark rule in experimental studies and probably the default control strategy in most waferfabs due to its simplicity. The MBS rule defines a threshold policy that allows starting processing only if a certain number of jobs above a given threshold is present, otherwise the machine remains idle. The batch process is delayed until the given threshold is reached. Akcali et al. (2000) experimentally examine the performance of different loading and dispatching policies for BP operations of a semiconductor waferfab. They particularly study the effect of MBS control, pointing out that the average flow time is a convex function of the threshold value of the loading policy. On one hand, a too low threshold level decreases the BPM utilization and increases the CT dramatically. On the other hand, a threshold level that is too high significantly increases the CTs. They particularly examine the effect of thresholds in low volume products. Refer to Figure 38 depicting the MBS rule behavior.

[11] http://www.appliedmaterials.com

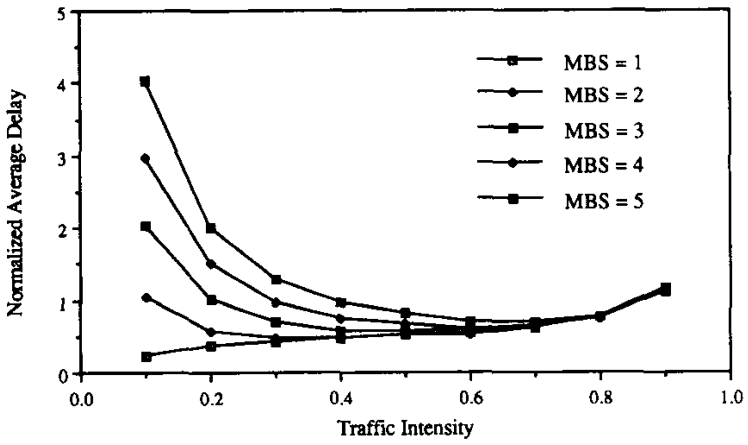


Figure 38: The MBS rule behavior (Glassey and Weng, 1991); cf. (Hopp and Spearman, 2001)

The optimal control policy for a single unbounded BPM, using only information about the current state of the system, is a MBS policy with an optimal threshold limit, which needs to be adjusted whenever system conditions change (Weiss, 1979; Robinson et al., 2000). Weiss (1979) presents an algorithm for finding the optimal control limit that minimizes the long run average waiting times. Avramidis et al. (1998) also develop computational procedures to minimize the expected long-run-average number of jobs in the system under threshold policies. Sung and Choung (1999) propose a neural network model that determines the optimal batch size in order to minimize CTs. Gurnani et al. (1991, 1992) compute the critical MBS threshold for a two stage serial batch system with the serial stage (e.g. photolithography) feeding the batch stage (e.g. furnace). Aalto (2000) shows that MBS control is the optimal operating policy for a single bounded BPM. Fowler et al. (2002) develop closed-form formulas used to determine optimum batch sizes for a system with multiple BPMs and multiple incompatible products. They propose a GA to find an optimum batch size for all products such that the total expected CT of any item is minimized. Phojanamongkolkij et al. (2002) and Phojanamongkolkij and Ghrayeb (2005) propose the use of an analytical queuing model to determine MBS thresholds in order minimize the sum of the weighted CTs of multiple products with different weights.

6.8.2 Look-Ahead Strategies

Modern waferfabs manage large amounts of information about the current state of the system, and thus principally provide capabilities to predict future arrivals with certain accuracy to a certain extend (Robinson et al., 2000). The Dynamic Batching Heuristic (DBH) is considered as the first published control policy that exploits the knowledge of future arrivals with the intention to reduce CTs at BPMs. Even in the presence of prediction errors incorporating future arrivals into the decision process leads to considerable improvements (Glassey and Weng, 1991; Robinson et al., 2000; van der Zee, 2007; Tajan et al., 2011; Kohn and Rose, 2013).

Section 8.2.2 and Section 8.2.3 discuss experiments that investigate the effect of predicted job arrivals in a scheduling scenario.

Dynamic Batching Heuristic (DBH) Glassey and Weng (1991) demonstrate that the use of forecasting information reduces the average CT of lots arriving at a BPM. Compared to MBS, the improvement depends on the traffic intensity. The observed improvement under extreme conditions (very low or very high traffic) is nearly zero, because DBH then operates like MBS in a sense that it starts only one lot in very light traffic and max batch sizes in very heavy traffic. In contrast, CTs can be reduced about 50% in moderate traffic scenarios (30% to 70% utilization).

Next Arrival Control Heuristic (NACH) Fowler et al. (1992a,b) propose the Next Arrival

Control Heuristic (NACH), demonstrating how the knowledge of future arrivals can be used to control a BPM, studying the multi-product case in particular. NACH, as well as all its variants, is aimed on minimizing the CT and only considers the next arrival. Fowler et al. (2000) further develop NACH in order to be applicable in the more general multiple products-multiple server case. Solomon et al. (2002) extend NACH in a sense that it additionally incorporates the setup status of downstream machines into the decision process. Another variant of NACH tries to control the inventory coming into the batch operation, demonstrating the benefit of pulling the jobs from upstream operations based on the needs at the batch operations. The idea is to control the feeding workstation with the goal that the batches arrive at the batch operation just in time so that unnecessary waiting time is reduced (Ham and Fowler, 2008).

Minimum Cost Rate Heuristic (MCR) Weng and Leachman (1993) describe the Minimum Cost Rate Heuristic (MCR) as an extension of the DBH, applied for the multi-product case on a single BPM. The major difference between DBH and MCR is the chosen look-ahead horizon. The DBH uses a fixed duration that limits the look-ahead horizon, i.e. the processing time. The MCR heuristic uses the processing time plus any prior waiting time as the scheduling horizon. Their experiment shows that MCR outperforms NACH, DBH, and MBS with respect to the average total queue length in the multiple product case.

Rolling Horizon Cost Rate (RHCR) Robinson et al. (1995) present the Rolling Horizon Cost Rate (RHCR), a control strategy that uses both upstream and downstream information. Their results confirm that significant improvements in CT can be realized by exploiting upstream information.

Dynamic Job Assignment Heuristic (DJAH)/Dynamic Scheduling Heuristic (DSH) The look-ahead strategy Dynamic Job Assignment Heuristic (DJAH) presented in (van der Zee et al., 1997) deals with multiple types of products being processed by a number of identical machines, additionally accounting for setup costs for a machine. The goal is to minimize the average CT per product in the long run. DJAH is then further extended to the online scheduling approach Dynamic Scheduling Heuristic (DSH); cf. (van der Zee, 2001; van der Zee et al., 2001). The proposed DSH creates a schedule for unrelated parallel machines, instead of just focusing on the machines available at the decision moment as discussed in previous studies. DJAH is also adapted to to control burn-in ovens in semiconductor manufacturing (van der Zee, 2004) and presented with an extension that is capable to manage non-identical job sizes (van der Zee, 2007).

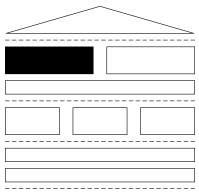
Model Predictive Control (MPC) Tajan et al. (2008, 2011) propose an online heuristic-based on Model Predictive Control (MPC) in order to reduce CTs for a single BPM with incompatible families. Their experiments show that the MPC-approach outperforms NACH in the multi-product case with respect to CT improvements. They also point out the potential benefits of controlling the upstream machines in a sense that an increased correlation in the job families is a way to effectively reduce CTs. Tajan et al. (2012) extend the MPC approach for a parallel machine environment.

Time-Limited Next Arrival (TLNA) Murray et al. (2008) present the Time-Limited Next Arrival (TLNA) heuristic that controls a set of parallel BPMs with setup times. TLNA takes future arrivals into account and aims on setup reduction, while trying to minimize a cost function consisting of the two conflicting performance measures total item queuing time and total machine running time.

Batch Apparent Tardiness Cost (BATC) Vepsalainen and Morton (1987) develop the ATC rule for minimizing the weighted tardiness in job shops. Several authors extended the ATC rule in order to solve batch scheduling problems, whereby the proposed rules are called BATC rules. See Section 6.5.2.

Look Ahead Batching (LAB) Gupta et al. (2004); Gupta and Sivakumar (2006) propose a Look-Ahead Batching (LAB) method demonstrating the benefit of using knowledge about the arrival times and due dates of future coming lots for the single BPM scheduling problem. The goal is to minimize three due date objectives simultaneously: earliness, tardiness, and their square sum. The steady-state simulation results show that exploiting the knowledge of future arrivals and their due dates leads to a significant reduction in the earliness/tardiness measures for tight and loose due date settings at two different utilization levels. Results have shown a significant reduction in these due date performance measures, especially for low traffic intensities.

More Real-Time Control Strategies Sha et al. (2004, 2007) develop a due date oriented look-ahead batching rule (LBCR) that considers the due date in order to raise delivery rates and reduce the average tardiness. They evaluate the proposed LBCR rule in a parallel machine environment, comparing it with other batching rules. Sahraeian et al. (2014) examine the ERT (Equalization of Runout Time) rule in order to minimize the C_{max} in a parallel machine environment with size dependent setup times and release times. Duenyas and Neale (1997) propose a batching heuristic applicable for a single BPM with random processing times in order to minimize the long-run average cost per unit time. They examine the static case where all jobs are available simultaneously and the dynamic case with job arrivals. Cigolini et al. (2002) propose a look-ahead procedure for scheduling several products on parallel batching machines.



7 The Framework

Contents

This section gives a brief description of the implemented framework designed to enable two main purposes: *a*) operational scheduling on the shop floor and *b*) exhaustive studies in an experimental system.

It provides a top level description of data systems, data transfer mechanisms and essential data procedures. The focus lies on data management —the extracting and transferring of data from and between scheduling sub systems.

The structure of this section is as follows: the core functionalities encapsulated in a structure of separated code packages and their basic interactions are described in Section 7.1. Section 7.2 investigates the VNS schemes implemented as an essential part of the framework. The framework's design stays abreast of two use cases: *a*) the use as an experimental system answering methodical questions and *b*) the use as a prototype to be deployed in the industry. Section 7.4 contains a description of the prototype and Section 7.3 describes the experimental system. Both subsystems share essential components of the framework despite of their differences that require case-specific adjustments in the written code.

Similar Frameworks Deployed in Industry The research literature provides a considerable number of descriptions of scheduling systems that have been successfully introduced in manufacturing systems such as waferfabs. The majority of those systems is based on simulation (DES) and makes use of an optimization method, e.g. heuristics or mathematical programming.

Liao et al. (1996) present the development of a daily scheduling tool based on IP for a R&D waferfab pilot line. Some general insights gained from developing scheduling systems for the industry are presented in (van der Krogt et al., 2009). More recently Bixby et al. (2006) and Fordyce et al. (2008) describe their efforts in developing and deploying a scheduling system in a 300 mm IBM waferfab.

Other reports only focus on parts of the manufacturing system. For example, Yugma et al. (2008) and Yurtsever et al. (2009) describe successful developments of batch scheduling systems in the diffusion area. In addition, many approaches target the scheduling problem in the backend of semiconductor manufacturers. Sivakumar (1999) describes a simulation-based scheduling system implemented in a semiconductor backend site. Similarly Horn (2008) presents a scheduling system based on DES for a backend-site; cf. (Weigert et al., 2009). Potoradi et al. (2002) describe the development of a DES system to a schedule weekly production in the assembly plant of a major semiconductor manufacturer.

Persson et al. (2006) present a successful application of simulation-based multi-objective optimization of another complex real-world scheduling problem that does not stem from wafer fabrication. See (Pinedo, 2008) for more examples of successful implementations of scheduling systems; cf. (Mönch et al., 2011a).

Similar Frameworks as Prototypes Beyond the success stories about scheduling systems successfully implemented in the real world, the research literature provides many promising descriptions of prototypes and concepts for scheduling systems designed to schedule waferfabs or parts of it. See (Gupta and Sivakumar, 2002; Habenicht and Mönch, 2003; Fowler et al., 2003; Mönch et al., 2003; Ham et al., 2009; Klemmt, 2012; Qiao et al., 2012a)

For more concepts of scheduling systems that are not explicitly designed to work in a waferfab but contain fruitful points of references in the area of scheduling complex facilities see (Wu and

Wysk, 1989; Smith et al., 1994; Drake and Smith, 1996; Chong et al., 2003a,b; Kumar and Nottestad, 2006; Monfared and Yang, 2007; Dangelmaier et al., 2006, 2007)

7.1 Code Packages

The framework primarily consists of a simulation-based optimization system developed to solve scheduling problems with BPMs in waferfab frontends. The underlying system covers various BPM scheduling problems that differ in their sets of constraints and objectives. An implementation of a generalized concept of VNS offers numerous search variants, including most of those mentioned in the related literature. The difference between these variants is basically the balance between exploitation and exploration of the search space, beside countless different parameter combinations to choose.

The framework is purposefully designed and implemented in order to operate two use cases: *a*) providing a proper environment to evaluate scheduling methods in experimental studies and *b*) offering the capability to serve as a prototype on the operational level of manufacturing. The first use case is to provide an experimental framework that offers to evaluate methods and problems under various settings comfortably. The latter satisfies industrial needs in terms of a functioning prototype that is fully connected to and properly operating with the waferfab's MES. Both use cases need different data structures and specific code routines managing their specialties. The core of the framework is a simulation-based scheduling system with optimization capabilities powered by VNS.

The framework basically consists of two main layers: a logic layer and a data layer. On one hand it covers the source code for the simulation-based scheduling engine with all its components realizing the modeling, simulation, and optimization tasks. On the other hand it comprises the underlying data infrastructure comprising the database, raw data preprocessing, data transferring, model loading and storing functionalities.

The developed scheduling system attains the status of a prototype on an operational level, providing the capability to load and validate a currently existing problem instance with actual data from the waferfab's databases. After loading a snapshot with actual data from the MES the scheduling procedure creates an improved schedule that in turn is written back to the MES for execution. See Section 7.4 for a description of the prototype.

In addition to implemented real-world and real-time features a model generator offers to create user-defined model instances of scheduling problems with specific characteristics. A database establishes the data management that is necessary to run the experimental system in an effective and comfortable fashion. Since large simulation/optimization experiments often suffer from a lack of computing power and time availability, the framework delegates extensive studies to a High Performance Computing (HPC) cluster with 64 cores connected to the database. The entire system is written in C#, with a larger focus on code comprehensibility than on computation speed. See Section 7.3 that describes the experimental system.

The core functionalities are designed and implemented with the ulterior motive to be used for both use cases, i.e. for the experimental system and the prototype.

The framework is organized in 10 main code packages: *a*) *DISPATCHING*, *b*) *EXPERIMENTATION*, *c*) *PROTOTYPE*, *d*) *MODELING*, *e*) *GUI*, *f*) *OBJECTIVES*, *g*) *VNS*, *h*) *SCHEDULER*, *i*) *SIMULATION*, and *j*) *SUPPORT*. The packages *DISPATCHING*, *MODELING*, *OBJECTIVES*, *VNS*, *SCHEDULER*, and *SIMULATION* are used in both use cases whereas the package *EXPERIMENTATION* only covers functionalities with experimental background and the package *PROTOTYPE* exclusively covers the prototype designed for the industrial application. Refer to Figure 39 for the package view of the framework.

Package *DISPATCHING* The package *DISPATCHING*contains a selection of implementations of well known dispatching rules. The dispatcher mainly operates on the model instance (covered with the package *MODELING*), simply comparing attributes of jobs in order to create ordered job lists. Refer to Section 6.5.1 for a brief introduction to dispatching rules.

Package *EXPERIMENTATION* The package *EXPERIMENTATION* comprises the definitions of every experiment described in Section 8. It defines each single run including the method

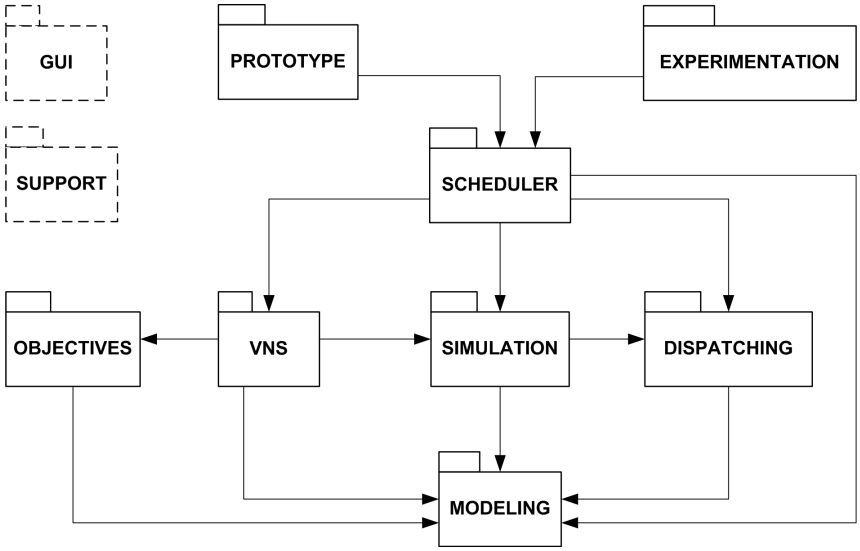


Figure 39: The package view of the framework

settings and the model instances, which are either loaded from the benchmark model or provided by the model generator. This package also provides the functionality that transfers the model instances as well as the running descriptions for each run of an experiment to the database. See Section 7.3 for a more detailed description of the experimental system.

Package *PROTOTYPE* The package *PROTOTYPE* contains a number of routines especially designed to satisfy the requirements for an industrial prototype. It fetches the data from the database and creates the model instance that mirrors the real world at that time instant. It triggers the scheduling engine and manages transferring the final schedule back to the database. See Section 7.4 for a more detailed description of the prototype.

Package *MODELING* The package *MODELING* contains the elementary objects such as machines and jobs among others, forming a model instance of a particular scheduling problem. This package covers the code core of the entire framework. Most of the other packages operate on the model instance defined in the package *MODELING*. The framework provides two options for creating a model instance: *a*) generating a model instance from experiment descriptions and *b*) loading a model instance from a snapshot of real-world data.

If the model instance is loaded with real-world data, a validating procedure will make the model instance validate itself. The model instance offers to be serialized in order to be directly stored on a file system or in a database in its object-oriented form. As the core of the framework, the package *MODELING* provides an elaborate object-oriented architecture that covers the atomic model instance of a particular scheduling problem. It contains all the information framed by objects necessary to create and optimize a schedule for a given problem. Each object stands for a physical or logical entity in the manufacturing process and is somehow connected to another object.

The objects can be divided into two types: *a*) the objects that describe the facility (e.g. routes, operations, and machines) and *b*) the objects that describe the scenario (e.g. lots and jobs). The altogether initialized und properly linked objects represent a model instance on which the simulation engine and the optimization procedures operate. Figure 40 only describes a simplified representation of the source code, although gives a brief insight into the core of the framework.

Package *GUI* The Graphical User Interface (GUI) defined in the package *GUI* offers rudimentary capabilities to visualize results and exemplary data related to the experiments. It also provides graphical procedures that visualize aspects of model instances, e.g. the distribution of machine attributes or job attributes; cf. Section 7.3.2 for a brief treatise about model generation.

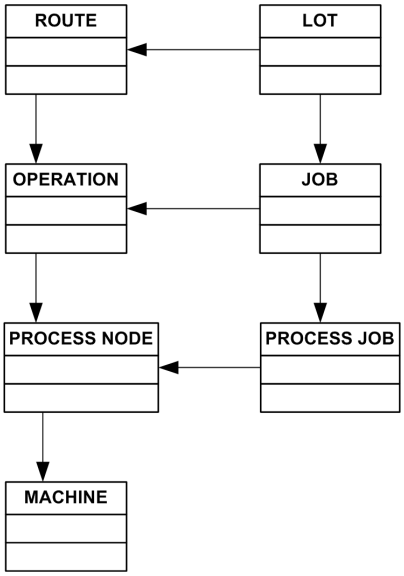


Figure 40: The simplified class diagram of the framework's package *MODELING*

Package *OBJECTIVES* The package *OBJECTIVES* frames multiple objective functions as well as several strategies that combine multiple measurements into one objective function. The multi-objective functions facilitate the hierarchical and the weighted linking of several objectives as well as the pareto scheme. This package also comprises an implementation of the Analytical Hierarchical Process (AHP), helping to identify proper weights for the subordinated objectives. See (Saaty, 1987, 1990, 1994) for the AHP and compare Section 1.2.4 for a brief review of the basic objectives that are typically subject of the optimization process.

Package *VNS* The package *VNS* contains all the functions related to the VNS scheme, comprising the implementations of the neighborhoods and the different search schemes described in the research literature. It further provides space for the search tracing components that when attached to a single run instance keep trace of every single move during the search procedure. See Section 7.2 for a more detailed description of the implemented VNS schemes.

Package *SCHEDULING* The package *SCHEDULING* represents a central code package of the framework. It combines the various packages and their functionalities to a powerful scheduling engine. The Time Window Decomposition (TWD) technique is used to decompose the model instance into a sequence of sub problems The Scheduling package combines all the subordinated packages that provide basic functionalities such as modeling, simulation and optimization. At the same time it serves as an interface to the top-level packages that manage the experimental system and the prototype.

Package *SIMULATION* The package *SIMULATION* contains the analytic models offering a fast equipment simulation capability. The main purpose is to simulate the processing of jobs on machines for short horizons. A time-based simulation concept transforms an ordered job queue in front of a particular machine into an exact job schedule with fixed start and completion times for the jobs. Refer to Section 4.7 for equipment modeling techniques.

Package *SUPPORT* The package *SUPPORT* provides functions and procedures often used in different packages. It basically deals with: *a*) data input and output procedures, *b*) mathematical functions, and *c*) visualization capabilities creating graphs and diagrams. The data input and output procedures provide the functionality to communicate with databases and operate on file systems in order to manage the data transfer. The mathematical functions are usually combinatorial

procedures and statistical functions. The implemented visualization procedures create graphs and diagrams representing complex data sets properly.

7.2 VNS Implementations

This simulation-based optimization framework employs VNS to create improved schedules with respect to the focused objectives. The concept of VNS, first described by Mladenović and Hansen (1997) and thereafter adapted by several researchers for a multitude of applications, proposes the definition of problem-specific neighborhood structures disassembling large scale optimization problems, i.e. COPs.

The basic idea behind is a systematic change of the neighborhood operator during the search, typically established in two alternating search phases: a descent LS phase and a randomized perturbation phase (shaking). See (Hansen and Mladenović, 2001, 2003; Hansen et al., 2001, 2009) for more detailed information.

The framework comprises an implementation of VNS as an abstraction of the originally proposed search schemes, which allows us to freely configure two nested search levels. Both levels can be parametrized independently from each other, where each search level defines a set of neighborhood structures, the LS procedure (first-improvement or best-improvement), and the shaking policy managing the shaking range (either constant or increasing).

Neighborhoods A neighborhood represents problem-specific knowledge by defining a certain kind of modification applied to a certain solution or parts of it. Each defined neighborhood constitutes a smaller partial problem offering the possibility to find improved solutions in an adequate time even for large COPs.

The neighborhood structure creates subspaces of the entire search space by encapsulating a certain set of operations used to modify the schedule. The search procedures in this framework are based on six neighborhoods: *a*) split a batch, *b*) merge two batches, *c*) swap two batches, *d*) move a batch, *e*) swap two jobs, and *f*) move a job.

Each neighborhood provides the capability to return a neighbored solution of another by defining LS procedures following the first-improvement or the best-improvement strategy. Section 8.3.2 discusses experiments that investigate the effect of the different neighborhoods in the search schemes and Section 8.3.3 deals with experiments investigating the two LS strategies embedded in different VNS variants.

The implemented neighborhoods are defined as follows:

Split a Batch The idea behind is to split a batch into two whereby one partial batch remains at its original position and the other one is assigned to another machine. Splitting a batch is simply achieved by defining a split index that refers to the job number in the batch. Any job below the split index remains at the original position in the original batch and the remaining jobs form a new batch assigned to a new machine.

Merge Two Batches Merging two batches means to transfer a number of jobs from one batch to another. It is even possible that two batches unite completely. The implementation simply considers the possible combinations of two batches, trying to move jobs from one batch to another.

Swap Two Batches Swapping two batches means that two batches exchange their positions in the schedule. This might imply that they even swap the machines they are assigned to. The procedure considers all the possible combinations of two batches, trying to swap their positions.

Move a Batch This neighborhood checks for each batch if another position on the schedule leads to a better schedule. The procedure simply tries to insert a certain batch at a new position on the schedule, even on another machine.

Swap Two Jobs Swapping jobs corresponds to the idea of swapping batches, but on the level of jobs. This neighborhood tries to exchange two jobs from different batches. This might imply that swapped jobs also change the machine.

Move a Job Moving a job simply refers to the transfer of a job from one batch to another. This is the neighborhood with the smallest impact on the entire schedule.

VNS Schemes This framework comprises the implementations of five basic variants of the VNS concept: *a*) VND, *b*) RVNS, *c*) BVNS, *d*) GVNS, and *e*) VNDS. Refer to (Hansen and Mladenović, 2001, 2003; Hansen et al., 2001, 2009).

VND and RVNS are considered as the basic building blocks, whereas BVNS, GVNS, and VNDS describe more sophisticated two-level compositions of them. VND repeats sequentially exploring neighborhood structures (searching for the best neighbor) of an incumbent solution until no longer an improvement is obtained. In contrast, RVNS does not perform LS; it randomly selects new neighbors changing the neighborhoods.

BVNS, GVNS and VNDS combine a LS scheme improving the incumbent solution with the ability to escape from local optima by the use of random movements in the solution space (shaking). Starting from an initial solution, a LS phase is continued until no longer an improvement is obtained. Without any knowledge about the optimal solution (global optimum), it must be assumed that the LS results always represent non-optimal solutions (local optima). The current solution is randomly modified in the shaking phase in order to escape from the local optimum. The shaking step tolerating deteriorations is subsequently followed by a LS procedure hopefully leading to a better solution.

The framework's implementation of the search schemes offers the option to use different neighborhoods for the shaking phase and for the LS phase. It is possible to define hundreds of different VNS search schemes by combining strategies and different parameter. The range of possible method settings covers deterministic variants that only employ LS as well as stochastic variants that manage to escape from local optima. These variants basically differ in the balance between exploring and exploiting search space. Additionally, the system supports multi-objective optimization, whereas multiple objectives are combined *a*) hierarchically, *b*) with weights, or *c*) equally in order to improve pareto fronts.

Since heuristic search procedures such as VNS operate on given solutions, it is required to provide an initial schedule as the start solution for each problem instance. The framework uses dispatching rules executed in a simulation system to generate initial schedules, which also provide the reference objective measures for analyzing improvements gained by optimization in the aftermath.

7.2.1 Variable Neighborhood Descent (VND)

The VND search method describes a neighborhood search scheme that changes neighborhoods in a deterministic way without any random effects. VND search differs from classical LS procedures: it employs multiple neighborhoods for improving a solution instead of using a single operator. The idea behind is that using multiple neighborhood structures increases the probability to find a global optimum (Hansen et al., 2009).

VND search in its original form is based on the best-improvement LS scheme that is also known as the steepest descent heuristic. It basically consists of two steps cyclically performed until no longer an improvement is obtained: *a*) exhaustive LS around the incumbent solution and *b*) moving to the best neighbor.

VND search first explores the current neighborhood $N(x)$ of the current solution x and identifies the best neighbor x'. Second VND search determines whether x' improves x and thus how to proceed. If x' is better than x, the search will restart exploring the first neighborhood using the improved solution x' as x. If x' does not improves x, the search will continue with the next neighborhood. When implementing the neighborhoods it is important to bear the complexity of the neighborhood operations in mind because exploring some neighborhood structures might be very time-consuming for some instances (Hansen and Mladenović, 2003); cf. (Talbi, 2009).

The principle of the VND algorithm is visualized in Figure 41. See Figure 42 for the pseudocode; cf. (Hansen and Mladenović, 2001, 2003).

7.2.2 Reduced VNS (RVNS)

The LS component is often the most time-consuming ingredient of VNS variants. The RVNS waives LS and randomly creates solutions in increasingly far neighborhoods (Hansen and Mladenović,

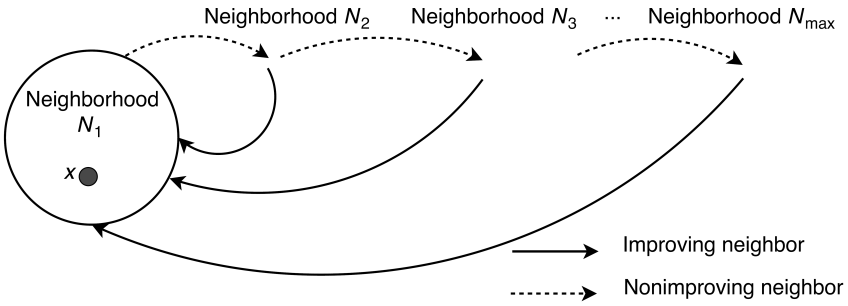


Figure 41: The principle of the variable neighborhood descent algorithm (Talbi, 2009)

Function VND (x, k_{max})
1 $k \leftarrow 1$
2 **repeat**
3 | $x' \leftarrow arg\min_{y \in N_k(x)} f(y)$ // Find the best neighbor in $N_k(x)$
4 | $x, k \leftarrow$ NeighborhoodChange (x, x', k) // Change neighborhood
until $k = k_{max}$
return x

Figure 42: The pseudocode for the VND algorithm (Hansen et al., 2009)

2001).

RVNS is originally designed to manage large valleys surrounding a local optimum. The original description assumes that the neighborhoods are nested, which means that the sequence of neighborhoods implies that each neighborhood contains the previous one.

RVNS basically consists of two steps cyclically performed until the stopping criterion is met: *a*) shake and *b*) move.

The procedure is as follows: Each cycle begins with selecting a new random solution x' around the incumbent solution x using the current neighborhood. If x' improves x, the search will be recentered and started again with the first neighborhood; otherwise the search proceeds to the next neighborhood. After all neighborhoods have been considered, the search restarts with randomly selecting a solution from the first neighborhood. The procedure is repeated until a stopping condition is satisfied (Hansen and Mladenović, 2003). It is recommended to use RVNS for very large instances for which LS is very costly (Hansen et al., 2009).

Refer to Figure 43 for the pseudocode; cf. (Hansen and Mladenović, 2003).

Function RVNS(x, k_{max}, t_{max})
1 **repeat**
2 | $k \leftarrow 1$
3 | **repeat**
4 | | $x' \leftarrow$ Shake(x, k)
5 | | $x, k \leftarrow$ NeighborhoodChange (x, x', k)
| **until** $k = k_{max}$
6 | $t \leftarrow$ CpuTime()
until $t > t_{max}$
return x

Figure 43: The pseudocode for the RVNS algorithm (Hansen et al., 2009)

7.2.3 The Basic VNS (BVNS)

The BVNS method combines deterministic and stochastic changes of neighborhoods. It enriches a LS procedure such as best-improvement LS with a shaking procedure that randomly selects solutions within the current neighborhood in order to escape from a local optimum obtained by the

LS scheme. Both the LS and the shaking step operate on the same neighborhood in every iteration, i.e. on the same search level. BVNS basically comprises three steps cyclically performed until the stopping criterion is met: *a*) shake, *b*) LS and *c*) move.

The procedure is as follows: Every cycle starts with randomly selecting a neighbored solution using the current neighborhood operating on the incumbent solution x. This is referred to as the shaking step, which leads to x'. Then the LS phase explores the result of the shaking step x', determining x'' that mirrors the best solution in the current neighborhood. If x'' improves x, the search is will be recentered with x'' and the procedure will begin again with the first neighborhood. Otherwise the cycle will proceed with the next neighborhood (Hansen et al., 2009).

See Figure 44 for the pseudocode; cf. (Hansen and Mladenović, 2001, 2003).

```
Function BVNS(x, k_max, t_max)
1  t ← 0
2  while t < t_max do
3  |  k ← 1
4  |  repeat
5  |  |  x' ← Shake(x, k)                  // Shaking
6  |  |  x'' ← BestImprovement(x')         // Local search
7  |  |  x, k ← NeighborhoodChange(x, x'', k)  // Change neighborhood
   |  until k = k_max
8  |  t ← CpuTime()
return x
```

Figure 44: The pseudocode for the BVNS algorithm (Hansen et al., 2009)

7.2.4 General VNS (GVNS)

GVNS combines VND with RVNS, which means that the resulting search scheme comprises a LS component as well as the shaking procedure. GVNS differs from BVNS in that the LS phase does not necessarily use the same neighborhood as the shaking phase in every cycle. The search is organized in two search levels; the first level performs the shaking procedure whereas the second level performs LS by VND. Each iteration is composed of three steps: *a*) shake, *b*) LS with VND and *c*) move.

First the initial solution x is shaked, meaning that a solution is randomly selected from the current neighborhood in the first level, resulting in solution x'. Then VND is used as a LS procedure in order to generate the solution x''. If x'' improves x, the search will be restarted with the first neighborhood. Otherwise the shaking procedure will try another random shaking move using the next neighborhood (Talbi, 2009).

Figure 45 shows the pseudocode; cf. (Hansen and Mladenović, 2003; Talbi, 2009).

```
Function GVNS (x, ℓ_max, k_max, t_max)
1  repeat
2  |  k ← 1
3  |  repeat
4  |  |  x' ← Shake(x, k)
5  |  |  x'' ← VND(x', ℓ_max)
6  |  |  x, k ← NeighborhoodChange(x, x'', k)
   |  until k = k_max
7  |  t ← CpuTime()
   until t > t_max
return x
```

Figure 45: The pseudocode for the GVNS algorithm (Hansen et al., 2009)

7.2.5 Variable Neighborhood Decomposition Search (VNDS)

The framework contains a variant of the VNDS adjusted for the focused use case so that the actual implementation differs from the original description of VNDS. The implemented variant of VNDS represents a two-level VNS scheme operating on two search levels that probably comprise two different neighborhood sequences similar to the described implementation of GVNS. The first level comprises a shaking step and a first-improving LS procedure. The second search level separately performs BVNS between the shaking phase and the first-improving local-search on the first level. VNDS basically comprises four steps cyclically performed until the stopping criterion is met: *a*) shake, *b*) BVNS, *c*) LS, and *d*) move.

The procedure is as follows: Every cycle starts with a shaking step on the incumbent solution x in order to create x'. On the second search level the BVNS procedure tries to improve x' leading to x''. The current cycle ends up with x''' as a result of the final first-improving LS step on the first search level. If x''' improves x, the search will be recentered with x''' and the procedure will begin again with the first neighborhood; otherwise the cycle will proceed with the next neighborhood (Hansen et al., 2009); cf. (Hansen and Mladenović, 2001; Hansen et al., 2001).

Refer to Figure 46 showing the pseudocode; cf. (Hansen and Mladenović, 2001, 2003).

Function VNDS $(x, k_{max}, t_{max}, t_d)$
1 **repeat**
2 $\quad k \leftarrow 1$
3 $\quad$ **repeat**
4 $\quad\quad x' \leftarrow$ `Shake` (x,k); $y \leftarrow x' \setminus x$
5 $\quad\quad y' \leftarrow$ `BVNS`(y,k,t_d); $x'' = (x' \setminus y) \cup y'$
6 $\quad\quad x''' \leftarrow$ `FirstImprovement`(x'')
7 $\quad\quad x,k \leftarrow$ `NeighborhoodChange`(x,x''',k)
$\quad$ **until** $k = k_{max}$
until $t > t_{max}$
return x

Figure 46: The pseudocode for the VNDS algorithm (Hansen et al., 2009)

7.3 The Experimental System

Designing and defining experiments as well as managing their execution is one of the central functionalities of this framework. The experimental system is purposefully designed and implemented to comfortably manage a remarkable amount of experiments. The driving force is to extend the framework with functionalities that make defining, executing and analyzing experiments as time-saving as possible. The experimental system basically comprises three components: *a*) the admin system, *b*) the database, and *c*) the HPC cluster.

The admin system initializes the database and the HPC machine, ensuring that their interplay results in a powerful experimental system that is capable of running thousands of experiments in a comfortable way. On one hand the admin system deploys the code of the scheduling engine to the HPC cluster and prepares its operating system for running a certain experiment. On the other hand the admin system transfers the experiment input data to the database from which the HPC machine requests its running descriptions. A model generator as a part of the admin system creates user-defined model instances with specific characteristics.

The database maintains the data management that is necessary to run the experimental system in an effective and comfortable fashion. It provides access to the experimental data that entails the input parameters including the model instances as well as the output results including the schedules. The database stores the descriptions of the experiments and the results of every single run. Each experiment usually consists of thousands of run descriptions related to the serialized model instances created with the model generator. The result of each run is at least a short summary of performance indicators derived from the created schedule. In addition it is possible to store the complete schedule with start and completion times of the jobs as well as the entire search trace with many additional information for each run. The experimental system is limited to model

instances that do not exceed 5 GB in their compressed size, which equals the maximum size for a Character Large Object (CLOB) in the database.

The HPC cluster offers the computing power needed to handle the large amount of experiment runs defined in the designs of the experiments. Since large simulation/optimization experiments often suffer from a lack of computing power and the time availability, the framework delegates extensive studies to a HPC cluster with 64 cores connected to the database. The HPC system is actually a DELL Blade Server with eight computing nodes, each with two Quad-Core E5450 Xeon CPUs (2,8Ghz) and 16GB memory per node, operated by Microsoft HPC Windows 2008 (64bit). Thus, the experiments were carried out on 64 cores running in parallel most of the time.

7.3.1 The Database Structure

The table structure comprises nine tables structured in two groups, i.e. input data and output data. The tables are hierarchically connected via primary and foreign keys, which enable a cascading deletion of related data sets in different tables. See Figure 47 for an overview.

Table *EXPERIMENT_ADMIN* The administration table *EXPERIMENT_ADMIN* serves as the anchor in the database. It contains the basic information of the experiment while connecting the input data tables and the output data tables.

Table *EXPERIMENT_INPUT* The table *EXPERIMENT_INPUT* only serves as an administration table that manages several input data sets for one experiment. The idea behind stems from an experience gained during developing and testing: in some cases it might be desired to run a particular experiment with another input data set generated with slight changes. The table is linked to the input data tables *EXPERIMENT_MODEL*, *EXPERIMENT_METHOD*, and *EXPERIMENT_BENCHMARK*.

Table *EXPERIMENT_MODEL* The (serialized) model instance with all its characterizing features is stored in the input data table *EXPERIMENT_MODEL*. It contains the serialized model object representing an instance of the scheduling problem. The table is rounded out with various information about this particular model instance, e.g. the number of machines and jobs among many others.

Table *EXPERIMENT_METHOD* The table *EXPERIMENT_METHOD* provides the settings for the scheduler. This table defines the method settings for each single run, e.g. the type of the initial solution and the maximum computational deadline among many other parameters refining the search scheme.

Table *EXPERIMENT_BENCHMARK* The benchmark instances used to evaluate the effect of different method settings on the search performance are stored in the table *EXPERIMENT_BENCHMARK*. It holds the serialized model instances of the benchmark as well as its characteristics and solutions for several methods with different settings.

Table *EXPERIMENT_OUTPUT* The table *EXPERIMENT_OUTPUT* only serves as an administration table that offers the opportunity to run a particular experiment more than once, e.g. with different versions of the scheduling engine. The experience in developing and testing a scheduling system advises to compare the results of different code versions based on identical benchmark instances. The table *EXPERIMENT_OUTPUT* is linked to the output data tables *EXPERIMENT_PERFORMANCE*, *EXPERIMENT_SCHEDULE*, and *EXPERIMENT_TRACE*.

Table *EXPERIMENT_PERFORMANCE* The table *EXPERIMENT_PERFORMANCE* receives a summary from every single scheduling run, including various KPIs derived from the schedule and some brief information about the optimization run, e.g. run times. Storing a summary with some analyzed KPIs for each run makes storing the entire schedule superfluous and thus saves space and time. The final result of each optimization run, the schedule, is only stored if needed.

Table *EXPERIMENT_SCHEDULE* The table *EXPERIMENT_SCHEDULE* collects the schedules from each scheduling instance if desired. If the schedules are available for a number of experiment's runs, it will be possible to derive more detailed information for the analysis, e.g. a specific distribution for the CT.

Table *EXPERIMENT_TRACE* The table *EXPERIMENT_TRACE* stores the data that describes the search behavior via traced search moves. Tracing a search is very memory-intensive since the search schemes perform many hundred thousand operations within a few minutes. The implemented search trace option provides insights into the search behavior and thus facilitates the development and the improvement of search schemes. See Section 8.3.6 for a short discussion about the search behavior of VNS.

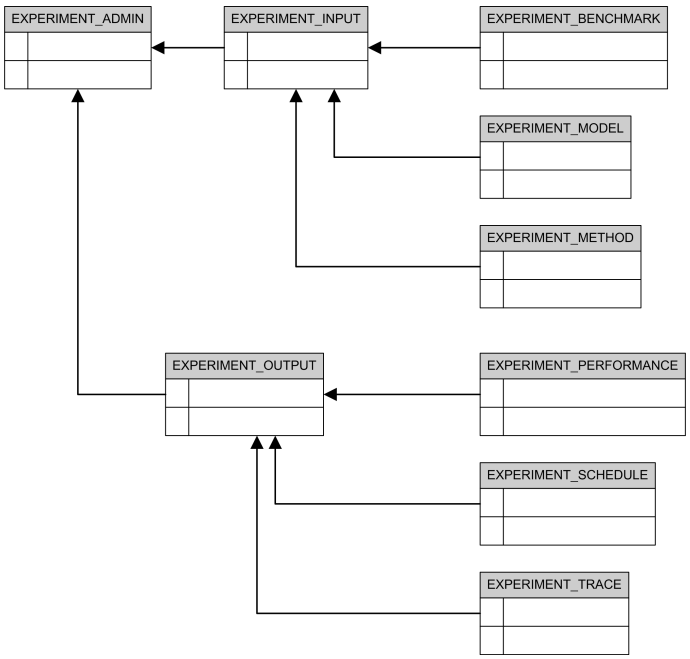


Figure 47: The data table structure of the experimental system

7.3.2 Model Generation

Generating models is one of the central functionalities provided by the experimental system that is part of the framework. The model generator is designed to create a set of independent model instances that show equal characteristics, but also differ slightly from each other. It sets the model variables using random numbers drawn from parametrized statistical distributions in order to create the model instances as defined in the experiment's description. Generating the model instances for an experiment is done in two phases: *a*) defining the model characteristics followed by *b*) creating the model instances as defined.

Defining the Model Instances There are numerous parameters that characterize a model instance and which need to be specified at the outset. Describing a model begins with setting the number of machines and jobs, and the utilization level if job arrivals are allowed. Then the description of the manufacturing facility is refined by defining the number of job families, the dedication scheme, the process times, the deadlines, and the batch sizes. It is noteworthy to say that the descriptions do not explicitly describe the later characteristic of every single machine but define the characteristic of the entire work center in terms of parametrized statistical distributions of machine characteristics. Setting the attributes of the jobs completes the description of an experiment's model instances. Just as with defining the machines, the characteristics of the jobs are drawn from parametrized statistical distributions that define the probability for a value for the

number of wafers, the priority class and its weight, the initial tardiness, and the arrival date. See Figure 48 that shows a snapshot from the framework visualizing the characteristics of a certain model instance.

Creating the Model Instances After dening the characteristics of the experiment's model, the creation of a defined number of independent model instances per model description begins. As dened in the model description, a set of independent model instances refers to a number of model instances with identical characteristics, but with a considerable number of differences on a more detailed model level. The differences among independent model instances stem from the fact that the model values on the machine and the job level are drawn from statistical distributions randomly created for every model instance individually. The point is that even if two statistical distributions are identical, e.g. both are normally distributed with identical average and variance, their actual values in the final array will not be. The experiments show that the differences among independent model instances created from identical descriptions can have a remarkable effect on the analyzed performance; cf. Section 8.2.1 and Section 8.2.2 for discussions about corresponding experiments.

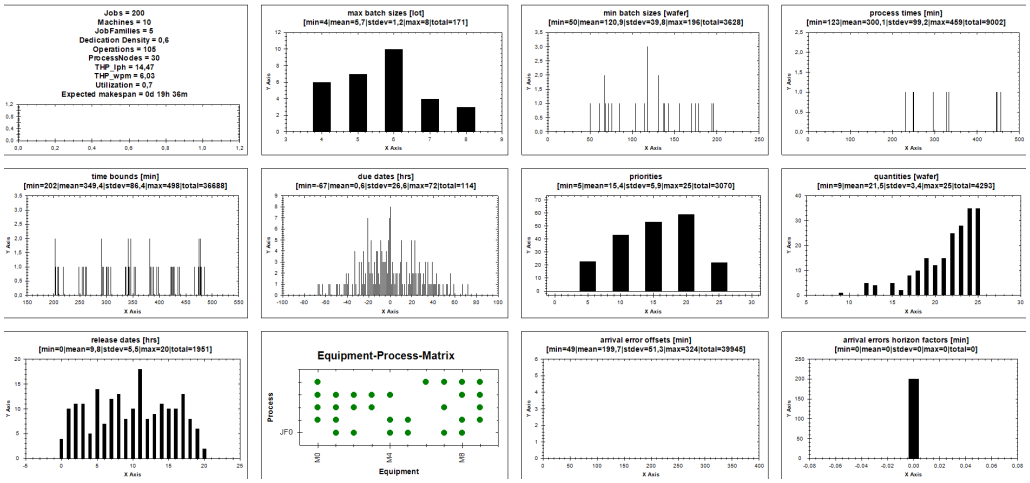


Figure 48: A snapshot from the framework visualizing the characteristics of a certain model instance

7.3.3 Administration and Data Flow

The admin system initializes the HPC cluster and transfers the experiment's input data to the database from which the HPC requests the running information for each of its runs. The HPC machine executes the optimization runs and stores the results in the database.

The communication between the HPC machine and the database works as follows: Whenever a computing core becomes free on the HPC machine, a new optimization run is triggered. Each optimization run proceeds with: *a*) retrieving the input data defined for the optimization run from the database, *b*) performing the optimization run with defined parameters, and *c*) writing back the results to the database.

See Figure 49 visualizing the data flow of the experimental system.

Initializing the HPC Machine The HPC machine performs the entire computation of all the optimization runs specified in the experiments. After deploying the code to the machine, the HPC manages an executable variant of the scheduling engine on all its cores and a plan of optimization run executions is defined. Every optimization run is initialized with another identifier used to request the run-specific information from the database, i.e. the model instance and method descriptions stored in the database for each run. The HPC operating system executes one optimization run after another, starting the next run whenever one of its 64 cores becomes free.

Transferring the Experiment The admin system comprises functionalities for defining the experiments as well as for transferring them to the database. Each optimization as a part of an experiment solves a particular model instance with a certain method. The number of optimization runs is equal to the number of models times the number of methods, given that every run is executed once (no replications). An experiment basically consists of two components: *a*) a number of model instances to be solved and *b*) a number of scheduling methods with specific settings. A remarkable amount of source codes implements functionalities for generating the model instances with specific characteristics for an experiment; cf. Section 7.3.2. Less but still considerable efforts have to be spent to manage different variants of the VNS schemes with alterable settings, which form the second component of an experiment beside the model instances. In contrast to the models every single method is exactly defined with all its parameters, e.g. the dispatching policy creating the initial solution, the distribution of the neighborhoods as well as the VNS scheme operating on these neighborhoods, and the stopping criterion in form of a maximum computation time or a maximum number of allowed moves.

Loading the Scheduling Model The optimization run instance requests its running information from the database, i.e. requests the model instance and the method settings. The model instance is stored and transferred in form of a string of characters representing the object-oriented model instance in a serialized form. The model loading procedures offer the capability to create an object model out of the transferred character string. This transformation procedure is also referred to as deserialization. The scheduling procedures begins after deserializing the model instance and initializing the VNS method as specified.

Executing the Optimization Run After loading the scheduling model, the scheduling engine first creates the initial solution using the specified dispatching rule and subsequently applies the VNS scheme described with the method settings. The VNS scheme optimizes the initial solution with respect to the objective function until the stopping criterion is met. The result of the optimization run is finally written back to the database.

Writing the Scheduling Results After executing the optimization run, the optimization run's result at least comprises a short summary of the KPIs derived from the schedule and sometimes the schedule itself. It is also possible to trace the search from beginning to the end, returning a complete documentation of all search moves. The results are then written back to the database for further analysis.

Analyzing the Scheduling Results Finally the admin system retrieves the experimental results for analysis from the database after the HPC machine performed the last optimization run defined in a particular experiment. The results stored in the database need to be processed by proper data queries in order to bring the data in a form that is suited to answer the questions stated by the experiment. Thus every experiment requires adapted data queries that make the results readable for analysis. The last step is to visualize the preprocessed experiment's data in diagrams that facilitate the analyzing and understanding of the experiment and its results.

7.4 The Prototype

Developing a prototype that captures the basic problem is an essential part of most technical research projects. An early prototype facilitates the ongoing research activities by providing practical experiences, acquiring the knowledge required to fully understand the problem with all its practical aspects. Both researchers and practitioners face the problem that the scheduling problems in the real world differ from the theoretical problem descriptions in academia. Hence practitioners are forced to develop customized solutions that finally fit to a specific real-world problem.

Another challenge is to integrate sophisticated scheduling solutions with the existing software structure, while generating a minimum of additional overhead. The amount of work that is needed to integrate new functionalities into a system is a decisive factor (van der Krogt et al., 2009). Fowler and Rose (2004) state that the development of real-time simulation/optimization systems

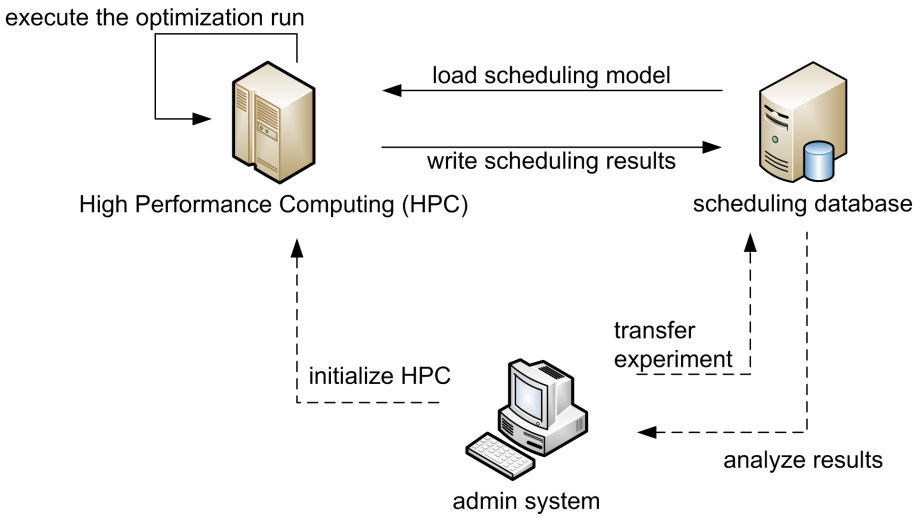


Figure 49: The data flow of the experimental system

as well as their plug-and-play interoperability with existing software is a grand challenge today. As a result of their experiences from launching a scheduling project, Fordyce et al. (2008) state that the biggest obstacle is no longer technology, but social order. Informally this means that the innovator trying to change the system will have those for his critics who are well off under the existing order of things.

The standard SEMI E105 (2000) gives a provisional specification of a scheduling component embedded in a manufacturing system. The specification roughly defines the interacting components, interfaces and required information to run a scheduling system. The scheduling component generally supports factory operations related to machines and material. It uses knowledge about product demand, equipment and material state, process flows, THP bottlenecks, operational policy, and many other types of information.

This section briefly describes the prototype as part of the framework with all its main components and functionalities. The first step in any project is to analyze the problem with its requirements. It begins with investigating the focused machines and their processes. Analyzing the latest active dispatching rule variant with all its special features reveals most of the requirements and constraints that need to be considered in a scheduling system. This prototype project is aimed at a small BPM work center with CFPs used for oxidation and diffusion processes. The scheduling problem is characterized with various constraints, e.g. machine eligibility constraints, time bounds, unequal job sizes, maximum batch size constraints (lots/wafer), and incompatible job families. The objective function usually comprises multiple components. For a detailed problem description see Section 1.2.

MRP, ERP, MES, and CIM Originally the idea of using computers for scheduling and inventory control is referred to as Material Requirements Planning (MRP). The MRP paradigm then evolved to Manufacturing Resources Planning (MRP II), which in turn merges into Enterprise Resource Planning (ERP). Finally the term ERP has emerged victorious as a synonym for any software controlling all company's operations, e.g. manufacturing, distribution, accounting, finances, and personnel. The MES can be seen as a software component within a MRP/ERP system. It automatically records, controls, triggers and executes manufacturing activities on the shop floor (Hopp and Spearman, 2001). Beside MRP and ERP the term CIM is often used.

Pinedo (2005, 2008) classifies information processing components of a manufacturing system, creating a hierarchy of components and discussing their relationships. The scheduling function takes place within an enterprise-wide information system that comprises a network of computers and databases connected with any type of equipment on the shop floor. Today the software that controls such an elaborate information system is typically referred to as an ERP system. See Figure 50 depicting an information flow diagram in a manufacturing system; cf. (Hopp and Spearman, 2001) for a discussion of components in a planning and control system.

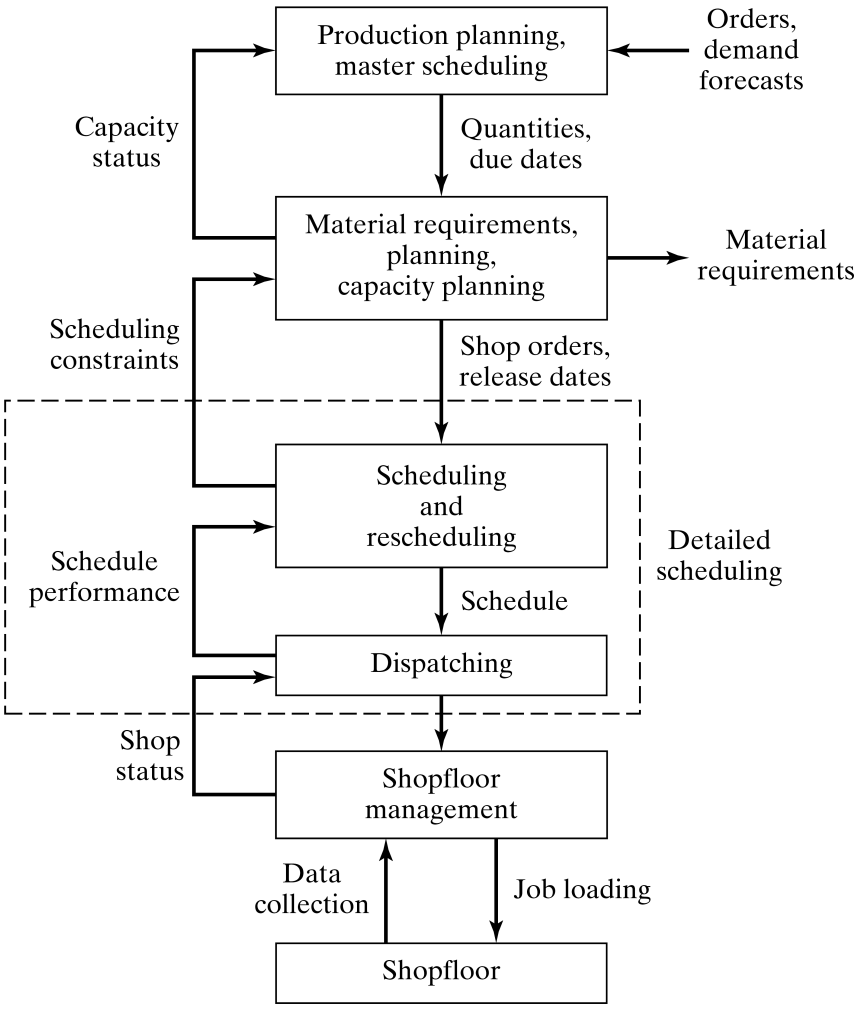


Figure 50: Information flow diagram in a manufacturing system (Pinedo, 2008)

7.4.1 Administration and Data Flow

Establishing the data transfer from the MES to the scheduling database is one of the most time-consuming activities during the development and the deployment of a scheduling system. The data that is required to create a model of the underlying scheduling problem is often distributed in many different data tables of the MES and other related databases. The high complexity of the data infrastructure in combination with the large volume of data leads to a considerable need of manpower developing software and managing data with all its problems. The data flow of the framework is based on dozens of data reports that extract, filter, analyze and merge data sets from different sources. This implementation of a prototype establishes a data flow between the three components: *a*) scheduling engine, *b*) scheduling database, and *c*) MES. See Figure 51 for a visualization of the information flow.

The scheduling engine comprises the simulation-based optimization system that generates improved schedules with the model data provided by the database. The scheduling database serves as a data storage and interface between the scheduling engine and the MES. It provides the model data that mirrors the situation on the shop floor as well as the optimized schedule generated to be executed. The MES in this concept refers to the data infrastructure that traces and controls the activities on the shop floor, i.e. provides updated input data for the scheduler and executes the schedule as proposed. The main data processing activities can be summarized in four groups: *a*) initializing the scheduling system (cf. Section 7.4.3), *b*) extracting the snapshot data (cf. Section 7.4.4), *c*) loading and validating the model (cf. Section 7.4.5), and *d*) generating and executing the schedule (cf. Section 7.4.6).

System initialization and administration frames all the activities necessary to prepare the scheduling system before the first schedule is generated, e.g. processing all the static data such as machine information and process routes. A very time-consuming part of the initialization procedure is fetching the static data (master data) from the MES, e.g. machine and process information. During the ongoing operations it is required to adjust settings and update input data in order to keep the system running as desired in view of changing circumstances. See Section 7.4.3 for a brief description of the system initialization routines.

The scheduling procedure itself comprises the last three steps cyclically performed in sequence: *a*) extracting the snapshot data (cf. Section 7.4.4), *b*) loading and validating the model (cf. Section 7.4.5), and *c*) generating and executing the schedule (cf. Section 7.4.6). The scheduling procedure continuously repeats the three actions in a cycle, resulting in a regularly updated schedule with high reliability if the interval is short.

The data extraction procedure fetches the required up-to-date information from the MES and transfers it to the designated scheduling database. The extracted real-time information represents the current status of the shop floor, including status information about machines and lots. The database completes the current real-time data with additional master data information needed for scheduling procedure, e.g. batch constraints and process times among many others. Additionally predictions such as job completion dates and job arrivals merge the current shop-floor status and the master data to the final model data that represents the scheduling problem. See Section 7.4.4 for a short description of the data extraction issues. The scheduling procedure finishes with generating an improved schedule using the model data and the optimization functionalities provided by the VNS implementation. See Section 7.4.5 for a brief introduction into the model loading and validating procedures. The schedule execution procedure operates independently from the scheduling procedure in asynchronous fashion. It repeatedly checks for the latest schedule as a part of the dispatching system, trying to realize it on the shop floor as proposed. See Section 7.4.6 for more details.

7.4.2 Data Structure

The massive amounts of data can be roughly divided into static data (master data) and dynamic data (snapshot data). The static data refers to information that changes rarely such as basic machine information or process flows. The dynamic data refers to objects in manufacturing that change their state more frequently such as lots.

The table structure (Figure 52) comprises twelve tables hierarchically structured in three groups and connected via primary and foreign keys, which enable a cascading deletion of related

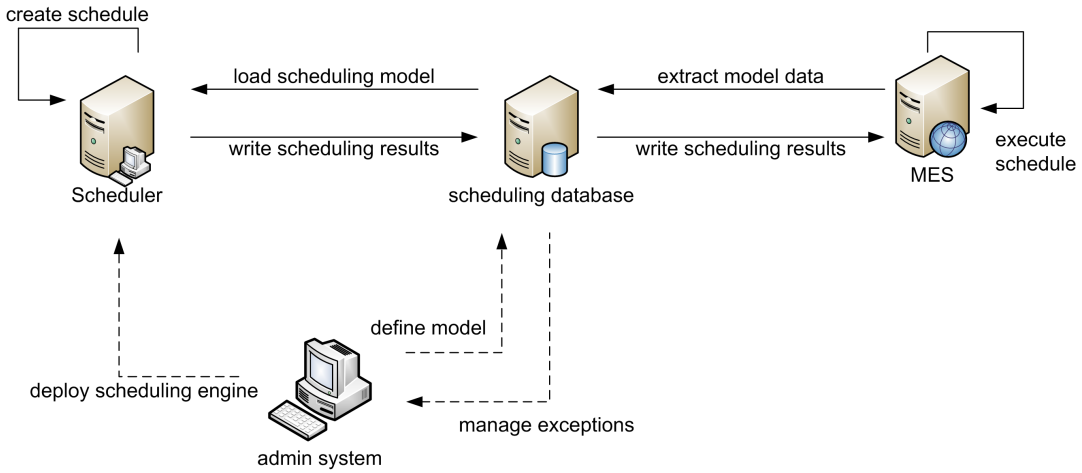


Figure 51: The data flow of the prototype

data sets in different tables. The three tables *MODEL_ADMIN*, *MASTERDATA_ADMIN* and *SNAPSHOT_ADMIN* are only used for organizational issues, where each table mirrors a data level. The root table *MODEL_ADMIN* on the first level serves as an anchor for the table *MASTERDATA_ADMIN* that in turn connects the table *SNAPSHOT_ADMIN* on the third level. See Figure 52 for an overview.

Making snapshots in short time intervals combined with the master data related to the machines and the process routes leads to a considerable amount of data, adding up to huge amounts of data when continuously running such a scheduling system day by day. The steadily increasing data volume requires an elaborated technique for deleting old data sets. A proper setting of primary and foreign keys ensures that the data tables remain consistent after carrying out clean-up activities. For example, deleting a model entry in the table *MODEL_ADMIN* results in deleting the related master data set(s) in the table *MASTERDATA_ADMIN* that in turn triggers a cascading deletion of all related snapshots in the table *SNAPSHOT_ADMIN*.

Model Data Level The model data level comprises four tables describing the model on the top level: *a*) table *MODEL_ADMIN*, *b*) table *MODEL_EQUIPMENT*, *c*) table *MODEL_ROUTES*, and *d*) table *MODEL_UPSTREAM*. The table *MODEL_ADMIN* provides the root entry for every model and the tables *MODEL_EQUIPMENT*, *MODEL_ROUTES*, and *MODEL_UPSTREAM* are directly connected with the root table *MODEL_ADMIN*, forming the administrative level of information describing the problem. The administrative data level simply lists the machines and the process routes that need to be considered for the focused work area; it basically defines the use case and its boundaries. Appendix D visualizes the boundaries of a exemplary model by showing the material flow around the focused furnace work center, showing the set of connected work center that sends to and receives material from the focused work center.

Master Data Level The master data level comprises three tables containing the static data: *a*) table *MASTERDATA_ADMIN*, *b*) table *MASTERDATA_PROCESS*, and *c*) table *MASTERDATA_TRANSPORT*. The table *MASTERDATA_ADMIN* orchestrates the master data organized in two tables: table *MASTERDATA_PROCESS* and table *MASTERDATA_TRANSPORT*. The table *MASTERDATA_PROCESS* provides all the data related to a process on a machine (process dedications, batch sizes, processing times, etc.). The table *MASTERDATA_TRANSPORT* stores estimated transport times needed to predict job arrivals. The tables *MASTERDATA_PROCESS* and *MASTERDATA_TRANSPORT* are linked to the table *MASTERDATA_ADMIN* that itself is linked to the table *MODEL_ADMIN*. This table structure offers to have one or more master data sets on the master data level for each model defined on the administrative data level. It is possible to let the master data or parts of it changing while the scheduler continuously operates without any synchronization problems.

Snapshot Data Level The third data level manages the snapshots; a snapshot refers to a data set that reflects the current state of the focused manufacturing system. The snapshot data level comprises five tables: *a*) table *SNAPSHOT_ADMIN*, *b*) table *SNAPSHOT_IN_STATUS_EQUIPMENT*, *c*) table *SNAPSHOT_IN_STATUS_LOT*, *d*) table *SNAPSHOT_OUT_SCHEDULE*, and *e*) table *SNAPSHOT_OUT_ERROR*. The snapshot data level has its anchor in the table $SNAPSHOT\_ADMIN$ that is connected to the table $MASTERDATA\_ADMIN$. This means that any snapshot belongs to a master data set that in turn belongs to a certain model. Vice versa, a model may have multiple master data sets where each master data set is related to several snapshots. Each snapshot contains the input data that is used to create the model instance feeding the scheduling routine. The input data provides information about the current states of the machines in the table $SNAPSHOT\_IN\_STATUS\_EQUIPMENT$ as well as about the states of the lots in the table $SNAPSHOT\_IN\_STATUS\_LOT$. Beside the input data, a snapshot is additionally linked with the output data resulting from the scheduling procedure, i.e. the generated schedule in the table $SNAPSHOT\_OUT\_SCHEDULE$ and the error messages in the table $SNAPSHOT\_OUT\_ERROR$.

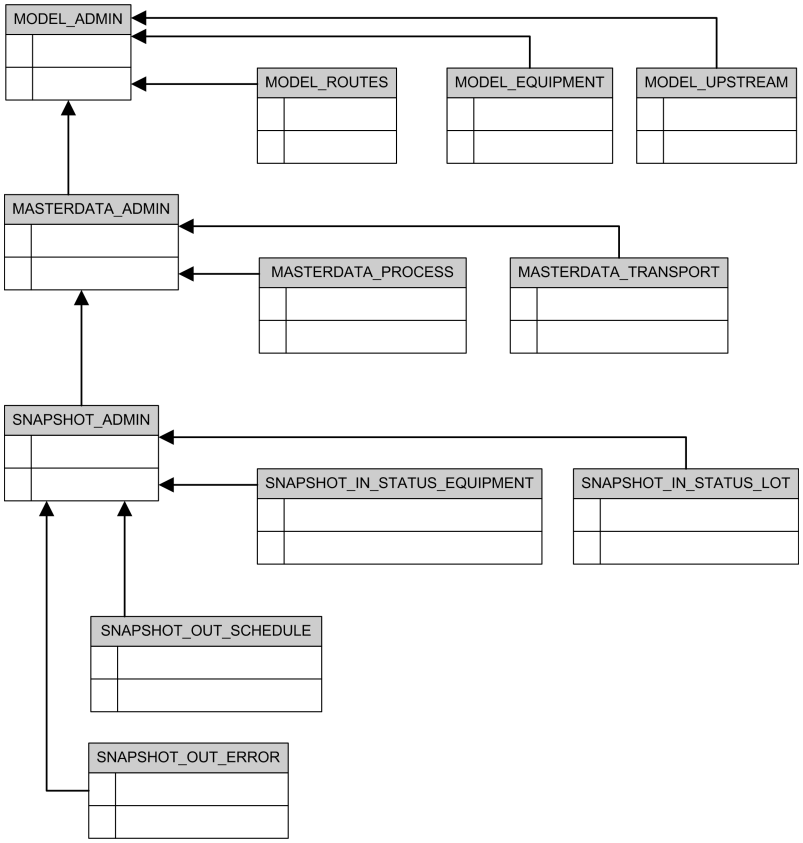


Figure 52: The data table structure of the prototype system

7.4.3 Initializing the Scheduling System

Initializing the scheduling system basically comprises two main activities: *a*) define the model boundaries and *b*) provide the master data.

Defining the Model Boundaries (Model Data Level) Preparing the scheduling system for operation begins with defining the boundaries of the model, e.g. determining the focused machines and the process routes. A list of machines defines the work center that is intended to be scheduled. The table $MODEL\_EQUIPMENT$ stores the work center's machine list for each model. A list of process routes implicitly defines the jobs to be scheduled. The process routes are stored in the table

MODEL_ROUTES with every single process step. The machine information of the focused work center combined with the process route information of the jobs not only define the work center but also build the basis for creating the upstream model that defines the preceding set of machines and processes.

The upstream model is used to create look-ahead information for predicting job arrivals to be incorporated into the scheduling decisions. The table *MODEL_UPSTREAM* contains the data that mirrors the relationships between processes and machines along the process routes.

Providing the Master Data (Master Data Level) Once the model with its boundaries is defined, the initialization phase of the scheduling system finishes with acquiring the master data that completes the model description. Processing the master data is usually very time-consuming because of the high level of technological complexity and the large amounts of data processed in a waferfab's MES. The master data corresponds to the machines and their processes, e.g. providing information about machine and process dedications, batch sizes, and processing times. The static data related to processes and machines is part of the master data level and stored in the table *MASTERDATA_PROCESS*. The table *MASTERDATA_TRANSPORT* provides transport times that estimate the time a lot needs to move from one equipment to another on average. The transport times mirror an important information for job arrival predictions.

7.4.4 Extracting the Snapshot Data

The scheduling procedure begins with ascertaining the status of the machines and the jobs. Their current status combined with the master data basically form the scheduling problem. A series of data reports is arranged in a kind of a master report managing the execution of the subordinated data reports in their intended order. The data reports preprocess the data in order provide a final snapshot data set from which the model instance for the current scheduling problem is generated. The master report with all its subordinated data reports gathers the data that is required to build the model instance for the scheduling problem at hand. The result of the snapshot is stored in the tables *SNAPSHOT_IN_STATUS_EQUIPMENT* and *SNAPSHOT_IN_STATUS_LOT* where the first table provides information about the machines (e.g. completion dates among others) and the latter supplies information regarding the jobs (e.g. arrival), as their names indicate. Creating a snapshot for a small work center (e.g. with five machines) approximately takes less than 2 minutes, including all the report-based data preprocessing activities and the transfer to the database.

Machine Status Report (Snapshot Level) The actual machine status, e.g. the machine is currently idle or processing, is a central information for the scheduler. The machine status report covers the machines that belong to the focused work center as well as to the upstream machines (all the machines that produce jobs having their next process on one of the machines in the focused work center). The table *SNAPSHOT_IN_STATUS_EQUIPMENT* provides the machine-related status information including the prediction of the completion dates of currently running jobs.

Job Status Report (Snapshot Level) The table *SNAPSHOT_IN_STATUS_LOT* offers place for all the job-related data, basically mirroring the current dispatching list showing the jobs available for processing. It contains all the jobs that need to be scheduled on the focused work center, i.e. the jobs already waiting for the process and the jobs that are currently in process at an upstream machine. The table *SNAPSHOT_IN_STATUS_LOT* is completed with master data such as batch constraints and processing times, which are essential for creating and optimizing a schedule.

7.4.5 Loading and Validating the Model

Based on the extracted snapshot data a model is created. Creating a model instance from the real world comprises three activities: *a*) loading the model, *b*) validating the model, and *c*) reporting validation results.

Loading the Model The task is to create an object-oriented model instance from a relational database consisting of a number of tables with distributed model data. The model loading procedure creates a number of objects from the database and links them together. The objects properly connected with each other mirror the underlying scheduling problem and provide the basis on which the scheduling engine operates. The model loading procedure computes the data row by row and table by table, where every row corresponds to at least one object. It is ensured that each object is unique, initialized with all the information provided by the raw data stored in the tables. The result is a model instance that contains a number of inter-connected model objects, e.g. jobs, machines, and processes. The model loading procedure merges the master data and the snapshot data into one model instance. Here the problem arises that both usually stem from different data reports executed at different times and thereby probably mirroring partially different states of the fab. When combining these different data sources it cannot be ruled out that problems occur. Thus a procedure needs to be implemented that validates the model in order to ensure an error-free model instance for the scheduling engine.

Three Main Sources of Typical Data Problems The extracted snapshot data in the database does not always exactly represent the status of the shop floor. The real-world data sets are subject to errors and thus validation procedures need to be installed in order to derive valid model instances from the data. Data errors in a typical scheduling project can be mapped into one of the three classified error sources: *a*) the complexity of the data infrastructure, *b*) the human element, and *c*) excursions on the shop floor.

a) The complexity of the data infrastructure with many interacting data systems causes that executing data reports comes with errors that lead to unwanted effects. Preprocessing the data involves several subsystems, multiple databases and various preprocessing routines operating on different data levels and system levels in order to provide the data as specified. If one subsystem does not function properly, the data sets will have errors, e.g. will show missing attributes or rows.

b) The human element is another source of erroneous data. Every piece of written source code operating on the data and manipulating it is erroneous in a sense that it is not functioning as desired in every situation, which might lead to serious problems. However, most of these errors in development remain undetected and will never show up in a critical situation. The data complexity in a waferfab's MES leads to a situation in which data experts need to spend much effort to gain the experience and the knowledge that is required to correctly interpret the data mirrored by millions of rows stored in dozens of data tables with several hundred attributes. As a consequence it is simply not possible to test every single situation that could occur. One can say that the data maintained by humans is always faulty, e.g. not kept up-to-date in every case. For example, it is not very uncommon that data policies lack proper data cleaning procedures, e.g. a specific data set created some time for a specific purpose but never deleted after its use ended. Thus databases often contain so-called dead data entries for which no one cares, but which can impede data analysis and its valid application.

c) In addition excursions on the shop floor such as process failures lead to erroneous data sets. Machine failures can produce unusual sequences of events that finally correspond to unusual data sets. An input data set that is different than expected can lead to erroneous data sets generated by the data preprocessing routines.

Validating the Model Data problems are considered to be a major barrier for transferring results from academia into a real-world application. Therefore researchers focus on techniques related to data validation and data mining in order to deal with missing or erroneous data (Mönch et al., 2011a,b).

This framework uses an object-oriented data validation procedure, ensuring that the input data satisfies necessary data quality standards. Validating the model instance follows after the model loading procedure is completed. Since faulty data leads to faulty objects, it must be ensured that the created model instance is free of any errors. This is important because creating a valid schedule that is ready for execution on the shop floor clearly requires valid model instances first.

But detecting errors and outlier is a complicated issue. In the first line it is required to define whether an object is valid or not. The validity of an object in this framework not only depends on its own attributes but also on the status of its related objects. It turns out that an arbitrary object as part of the model instance can be considered as valid if there is no invalid object related to it.

The model loading procedure first ensures that each object itself is considered as valid and second that it is exclusively linked to valid objects. The validating procedure employs rules that define the acceptable range of attributes, finally defining valid or invalid objects. Similar rules exist for the relationship between the objects. These rules (implemented by every object) enable each object to validate itself by determining its validity and undertaking proper actions (such as removing references to invalid objects) that can restore its validity. The validation procedure cyclically checks all the objects of their validity until no more invalid object is detected. The objects detected as invalid are removed in every cycle and so are their references from other objects. Removing invalid objects and references might lead to fragmented objects that do not play a role in the model instance because they are isolated in a sense that they do not show any reference to another object; thus isolated objects are removed. The validation procedure ensures that every valid object building up the model instance is characterized by attributes set to proper values. In addition valid attributes exclusively refer to other valid objects. Whenever an object must be removed the validating procedure records the reasons for its removal in order to identify the error sources in the preceding data preprocessing stages. The validation algorithm implemented in every object is roughly as follows: *a*) check whether all attributes were set to a value (no NULL entries), *b*) check whether the attribute values satisfy their validity constraints (e.g. a number lies within a given range), *c*) check whether the referenced objects satisfy the first two aspects (internal validity), and *d*) remove invalid references and restart the validation procedure if necessary.

Reporting Validation Results The validation report contains any removed object and a short description of the reasons that lead to its removal. In the aftermath it becomes possible to trace the errors by identifying the sources of the objects' invalidity. Such a reporting capability is indispensable when trying to ensure that valid model instances result from probably error-prone data. The table $SNAPSHOT\_OUT\_ERROR$ stores the results of the validation procedure.

7.4.6 Generating and Executing the Schedule

The initial solution is provided by a classical dispatching rule, e.g. a variant of BATC. The computational deadline is set in accordance to the size of the work center, which means that more machines recommend a higher computational time limit. See Section 8.3 for a discussion of the best method settings; it deals with experiments investigating numerous parameters.

The objective function bears in mind that several objectives play a role in practice. First and foremost the triplet of THP, CT and OTD is used in the first place, considering that jobs can have different weights mirroring their priority. Beyond these three it is additionally of particular importance to avoid time-bound violations. The question is how to combine these objectives. There are three basic strategies for combining multiple objectives: *a*) using a hierarchical structure, *b*) using weights, and *c*) using the pareto scheme; cf. (Kohn et al., 2013). By supporting the weighted variant, the AHP affords an appropriate opportunity to define a multi-objective function by determine the weights of the components.

The top priority goal is always to prevent infeasible solutions, e.g. in cause of violated time bounds. In the case that the scheduling procedure is not capable of finding a schedule without any violated time bound, the scheduler reports an error written to the table $SNAPSHOT\_OUT\_ERROR$. A warning report frequently checks whether the table $SNAPSHOT\_OUT\_ERROR$ contains any warnings that point to critical jobs, e.g. those that probably violate their time bounds in near future. A warning system is required to inform the responsible people on the shop floor in order to facilitate the exception handling. A transparent documentation of the errors helps determining and eliminating sources of such disturbing irregularities in the early phases of a scheduling project.

The result in form of a schedule is written to the table $SNAPSHOT\_OUT\_SCHEDULE$, which serves as a kind of data gateway to the MES where the schedule is then directly accessible on the shop floor. Transferring the optimized schedule to the MES constitutes the final step in the

entire scheduling procedure that subsequently restarts its cycle with extracting the data for the next snapshot.

Executing the schedule as created might expose problems. The reason is that the current schedule is based on a snapshot mirroring the state on the shop floor a few minutes ago. Unfortunately there is the danger that the status of the machines and the jobs have changed during the last scheduling cycle, which means that the proposed schedule is no longer executable as initially calculated. There exist various problems that might occur, e.g. the status of a machine has unexpectedly changed and is not available for processing any longer. An unexpected change of a job's status may also lead to an invalid schedule, e.g. when a predicted job is subject to some transport problems for example. Refer to Section 6.1 that points to the topic of rescheduling strategies and robust optimization.

The proposed schedules need to be properly communicated to the responsible experts in order to facilitate their believe in a functioning scheduling system. The visualization of the latest scheduling result via Gantt-charts remarkably facilitates the user acceptance, but only if the results appear reasonable. This framework uses a software developed from project partners in order to create Gantt-charts; cf. (Lange et al., 2009). See Figure 53 for an exemplary Gantt-chart generated with the prototype.

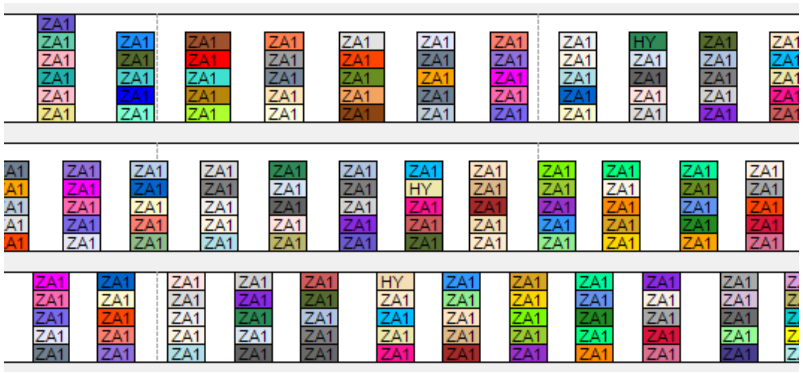
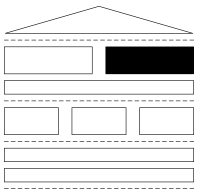


Figure 53: An exemplary Gantt-chart generated with the framework



8 Experimental Studies

Contents

One might think of four basic methodologies to evaluate the benefit of a scheduling system in comparison to a dispatching system: *a*) online evaluation in reality, *b*) offline evaluation in parallel with current real-world data, *c*) offline evaluation with historical real-world data, and *d*) offline evaluation via a simulation system.

Hopp and Spearman (2001) note that it is important to remember that a manufacturing system consists of equipment, logic, and human resources. Especially the human element in operations management needs to be considered when designing, implementing, and finally taking a scheduling system into operation. A functional system also provides a real benefit for the professionals who are dealing with it on a daily basis.

Online Evaluation in Reality A very straightforward strategy is to implement and deploy a scheduling system and to compare the performance of the work center in focus before and after the deployment. The problem with this strategy is that two dierent time frames in production are very rarely comparable due to a multitude of disturbing inuences impacting on the performance of a system because of the situation continuously changing on the shop oor. Therefore it cannot be ruled out that the two manufacturing situations (before and after) describe completely different production scenarios with different characteristics, e.g. with different utilization levels or product mixes. This involves the danger that a comparison of different time frames with different characteristics results in misleading conclusions concerning the benefit of the evaluation.

In addition start-up difficulties that typically occur in the early stages of system deployment can disparage the benet of an evaluation. Furthermore the process of developing and deploying a scheduling system is associated with high efforts and costs. Managers usually spare expenses especially when they are not convinced that the system will bring a return. Fowler and Rose (2004) argue that the biggest challenge today is to persuade the management to sponsor modeling and simulation projects instead of classic manufacturing methods such as lean manufacturing and six sigma; cf. (Halevi, 2001) for an overview on 110 manufacturing methods. However, evaluating a prototype system under real-world conditions is often the first choice since it generates the most reliable results compared to the alternatives.

Offline Evaluation in Parallel with Current Real-World Data One might think of another evaluation strategy: running the scheduling system offline in parallel without touching the dispatching system on the shop floor. The idea behind this is to compare the real-world decisions of the dispatching system with the virtual decisions of the scheduling system running parallel in the background in order to derive some measures that estimate the expected benefit. Unfortunately this strategy will not work properly because the offline scheduling system continuously needs to be synchronized with its dispatching pendant in the real world in order to create two aligned data sets. Otherwise the simulation will diverge from reality due to differing dispatching/scheduling decisions. On the other hand, aligning the scheduling data set means at the same time neglecting the virtual scheduling decisions and the result would not warrant a reliable analysis of benefits.

Offline Evaluation with Historical Real-World Data Another offline evaluation method is based on data sets from history. This requires technical capabilities to trace the production process with many details in order to reproduce the manufacturing history in the simulation system. This strategy faces two main problems. First the amount of historical data with the required level of detail is rather small, which leads to experimental results with less reliability compared to

experiments with —theoretically endless —amounts of generated data. The second problem is that, despite of a high level of details, not every activity that influenced the production in history can be properly traced. The lack of details leads to remarkable inaccuracies in the evaluation analysis.

Offline Evaluation via a Simulation System Another reasonable way to compare a dispatching system with a scheduling policy is to strictly use a simulation approach. Simulation systems enable reproducible experiments with stable and predictable conditions. Critics argue that a simulation system is too far away from reality and not enough shop floor specifics are considered.

Motivation for the Design of Experiments Despite of the widespread success of metaheuristics very little is known about their functioning, i.e. their interaction with the search space. The reasons why metaheuristics work so well (and under what conditions) remain unidentified to a large extend.

Actually many factors determine the performance of an algorithm. On one hand, the model clearly defines the range of the improvements. That is the reason why methods are usually compared with each other by using identical benchmark model instances solved by the methods to be compared. On the other hand, it is obvious that the nature of the search scheme, i.e. its algorithmic components and search features, determines the performance of the search method. But beyond these two important aspects, it is known that a search method's performance also depends on the fine tuning of the algorithm's control parameters and the realization of the actual implementation among other factors (Watson, 2009).

Hooker (1995) argues that the competitive testing paradigm that is present in the algorithmic research community is not suited to create that sort of insight that is required to develop more effective algorithms on the long run. The competitive testing paradigm describes the commonly accepted way algorithmic scientists present new findings that improve the state-of-the-art. Researchers typically focus on demonstrating the superior performance of their search scheme compared to another, rather than analyzing the algorithm's performance in order to find where the improvements stem from. Most often it is considered to be sufficient to show that one algorithm outperforms another, but not why (Watson, 2009). In the same line Potts and Strusevich (2009) state that a better understanding of the metaheuristics' mode of operation can provide the knowledge that is necessary to design high-performing search schemes.

Hansen and Mladenović (2003) mention promising areas of research that offer opportunities to improve the performance of VNS: *a*) initialization, *b*) inventory of neighborhoods, *c*) distribution of neighborhoods, *d*) ancillary tests, *e*) use of memory, *f*) parallel VNS, *g*) hybrids, *h*) using VNS within exact algorithms, *i*) artificial intelligence, *j*) enhancing graph theory with VNS, and *k*) solutions with bounds on the error. This chapter takes a closer look at a selection of these issues.

In addition to experimental studies that are somewhat detached from reality, it is necessary to design the studies staying in touch with the real-world fabrication environment. It is of great importance to analyze and to understand the practical situations in which operation planning and control strategies come into use. By thoroughly analyzing the problem in practice it is possible to describe the nature of important problem parameters such as processing times or important objectives. For example, it is inherently necessary to know about the maximum problem size encountered in practice (Gupta et al., 2006).

The experimental results are discussed in three sections. Section 8.1 deals with the static scheduling model without job arrivals, examining the effect of the very basic problem parameters that characterize a problem instance. Section 8.2 extends the model with dynamic job arrivals, covering important aspects that play a role when optimizing scheduling models with job arrivals. Section 8.3 puts focus on the method, evaluating various method settings on the basis of a set of benchmarks

8.1 Static Models

This section deals with experiments that investigate the relationship between model characteristics and the improvement in the objective gained by optimization. It investigates some interesting effects from selected view points chosen from the experimental results. The experiments in this section investigate the settings of the model instances in order to clarify the model-related circumstances

under which a certain level of improvement is observed. The idea behind this is the observation that optimization improvements depend on the model instance to a large extend, respectively on its characteristics.

The Default Settings for Model and Method The problem under study belongs to the class of parallel BPMs scheduling problems with incompatible job families. It can be seen as a scheduling problem with unrelated machines since the processing times for a job also depend on the actual machine. The processing time for a job is determined by the processing time of the related job family on a certain machine which means that the processing time for a distinct job family can differ on the machines.

The default model contains 8 machines and 120 jobs to schedule. None of the models in this section deal with job arrivals, i.e. all the jobs are available at time zero. For each design point, respectively for each parameter combination, 30 independent model instances are solved; there are 30 problem instances at each design point. Every single experiment is based on a modification of the default model, varying up to four parameters at the same time in order to study their relations to other factors and measures.

Generally VNS is a two-level search scheme combining a LS phase and a perturbation phase. The series of experiments discussed in this section exclusively employs VND as the search method, which deterministically leads to a local optimum. The method setting defines six neighborhoods: *a*) split a batch, *b*) merge two batches, *c*) swap two batches, *d*) move a batch, *e*) swap two jobs, and *f*) move a job. Refer to Section 7.2 for a detailed description of the implemented VNS variants.

The run time measure of the VND procedure serves as an estimate for the complexity of the problem, respectively for its size. It represents the time required by the VND search algorithm to identify a local optimum in the search space. VND terminates if no more modifications in any neighborhood lead to any further improvement. It is assumed that a longer VND run time refers to a larger search space generated by a problem instance of higher complexity. In another way, it is assumed that the number of moves that the algorithm requires to identify the first optimum corresponds to the size of the entire problem, respectively its complexity.

The maximum computing time is limited to a 10 minute maximum. The search always evaluates the objective value of the first optimum for all problem instances regardless of their size since VND operates deterministically. There is only one exception: those model instances for which VND is not capable of identifying a local optimum within the maximum deadline of 10 minutes. If the VND procedure is not finished within the 10 minute deadline, i.e. no local optimum was found in time, the algorithm stops and the current solution, which is the best known at this time, is returned.

VND can be considered as a deterministic (best improvement) LS strategy operating on a limited set of neighborhoods specifically designed to solve the BPM problem. There are two reasons justifying the decision to apply a deterministic VNS variant, instead of choosing a stochastic one, which has been proven to outperform deterministic approaches. On one hand the deterministic behavior of VND reduces the total number of experimental runs, since there is no need to run multiple replications that guarantee a certain level of statistical reliability, in contrast to stochastic VNS derivatives. On the other hand deterministic LS provides a better understanding of scheduling complexity with regard to the size of the problem instances. Examining measured computing times and/or number of search moves, combined with an analysis of improvements by optimization related to performance measures, clarifies whether local optima can be found for certain problems or not, in compliance to given computational deadlines.

The dispatching schemes SPT and EDD generate the initial solutions. The SPT rule is used when the objective is to minimize CT. The EDD rule is used when the experiment deals with OTD, i.e. the objective is to minimize tardiness, lateness, or unit penalties. The dispatching interval is set to five minutes, which means that every five minutes the dispatching procedure is executed, which probably results in a new job/batch started. The initial solution serves as a reference point for the optimization procedure at the same time, which means that the improvement is always given as the ratio of the values of the optimized solution to the dispatching solution. Refer to Table 21 for the default settings.

The Experiments This section describes the results of nine experiments, each taking a closer look at a certain factor or a set of factors.

Table 21: Default settings for the series of experiments related to the static model

factor		level
basic model parameter	machines	8
	jobs	120
machine parameter	job families	8
	dedication density	$1(uniform)$
	processing times	$U(240, 480)$
	batch size (lots)	8
	batch size (wafers)	200
job parameter	job sizes	$s_j = 25$
	job priorities	$w_j \in U(1, 5)$
	job due dates	$d_j \in r_j + N(12, 12)$
	job arrivals	$r_j = 0$
model instances per parameter combination		30
method parameter	initial solution	EDD (TT) / SPT (TCT)
	objective(s)	TCT/TT
	VNS type	VND
	deadline	10 min

The first experiment E.1 is focusing on the method run times that mirror the problem's complexity. Various factors determine the complexity of the problem, respectively its size. The experiment E.1's intention is to determine the computational complexity of certain problem instances quantitatively by measuring the time required to identify a local optimum by VND. The search method VND in the experiment E.1 uses a random schedule as initial solution and minimizes either C_{max}, TCT, TWCT, TT, or TWT. The experiment E.2 puts emphasis on the objectives TCT and TT, analyzing the improvement in the objectives and its dependency on other factors. The experiment E.3 is designed to study the effect of the batch size.

The experiments E.1, E.2 and E.3 investigate different combinations of numbers of machines, jobs, job families, as well as different process time schemes. The problem can be formally denoted with $Rm \mid p - batch, b < n, fmls \mid \cdot$. The models cover either 4, 8, 12, or 16 machines. Similarly there are four model variants with different numbers of jobs, i.e. 120, 160, 200, or 240 jobs; the jobs belong to 4, 8, 12, or 16 job families. Furthermore, the models differ in their processing time schemes: $U(120, 480)$, $U(240, 480)$, or $U(360, 480)$. For example, $U(240, 480)$ means that the processing times for the job families on the machines are uniformly distributed between 240 and 480 minutes. Especially the experiments E.1 and E.2 provide run time measurements that help to determine the computational boundaries of the experiments, creating some guidelines towards the design of the following experiments.

The experiment E.4 evaluates the effect of machine eligibility constraints, using models with different dedication density factors for the problem $Rm \mid p - batch, b < n, fmls, M_j \mid \cdot$. Another important constraint is examined in the experiment E.5 that is designed to study the effect of job sizes; the problem is denoted with $Rm \mid p - batch, B, s_j \mid \cdot$. The experiment E.6 analyzes the effect of proper due date settings and the experiment E.7 examines different number of job priority classes. The relationship between multiple objectives is the focus of the experiments E.8 and E.9. The experiment E.8 studies the correlation between the objectives, whereas the experiment E.9 provides results for bi-criteria optimization scenarios. The experiments E.6, E.7, E.8 and E.9 examine the problem $Rm \mid p - batch, b < n, fmls \mid \cdot$ with different objective functions.

All the model instances in the experiments E.3 to E.9 are designed to allow the VND algorithm to find a local optimum for the analyzed problem within the given time limit of 10 minutes. In contrast the experiments E.1 and E.2 contain model instances of larger sizes that prevent VND from finding a local optimum within 10 minutes. The appendix contains all information for each

experiment regarding the exact design of the experiment and the analyzing charts of improvements, run times and traced search moves.

Refer to Table 22 for an overview on the experiments carried out.

Table 22: Overview of the experiments related to the static model

experiment	studied factors	model instances	total runs
E.1	machines, jobs, families, process time	5760	28800
E.2	machines, jobs, families, process time	5760	11520
E.3	batch sizes, families, process time	4320	8640
E.4	families, dedication density	1200	2400
E.5	families, job sizes	960	1920
E.6	due dates	810	2430
E.7	priorities	750	1500
E.8	objectives (correlation)	360	2880
E.9	pareto-objectives	360	5400

8.1.1 Objective Function

The objective function as one of the most important factors characterizes a scheduling problem in a remarkable way and inherently determines its complexity. From the complexity theory it is known that some objectives are easier to minimize than others, e.g. minimizing C_{max} is easier than minimizing TT. This fact is also mirrored by the computing times. For example, the run time for C_{max} is strictly shorter than that for TT. Similarly, the computing times show that minimizing TCT is easier than minimizing TWCT. Strictly speaking, the experiments show that the VND search settles earlier into an optimum when minimizing TCT compared to the TWCT case. The experiments E.1, E.2 and E.6 provide scheduling results and run times for the problem $Rm \mid p-batch, b < n, fmls \mid \cdot$ with various objectives.

However the current state-of-the-art in complexity theory still leaves questions unanswered. For example, the results from the complexity theory do not provide a direct reduction path between C_{max} and TCT, which means that it is not prooven which objective is harder to solve. But when comparing the run times of TCT and C_{max} with each other, it becomes obvious that minimizing TCT clearly requires more time. This observation provides indication that problems with the TCT objective are harder to solve than those with the C_{max} objectives. The experiments also show that the run times for TWCT and TWT appear to be nearly identical. Refer to Figure 54, Figure 56, Figure 59, and Figure 61; cf. Appendix E.1.

For designing and running experiments it is of essential importance to own some key data about model sizes and run times of basic problems with typical objectives. The experiment E.2 shows the results for the objectives TCT and TT, compared to SPT and EDD dispatching. The run times for TCT stay below the maximum limit of 10 minutes in all cases, which means that VND is capable of finding a local optimum for all model instances within that time. VND identifies a local optimum for even the probably most complex design point with 16 machines and 240 jobs within 10 minutes. When analyzing the experiment E.2's results it becomes obvious that minimizing TT requires more time than minimizing TCT, confirming the results known from the complexity theory (TCT $\propto$ TT). It is easy to create a scenario in which searching 10 minutes is not enough to identify a first local optimum. VND finds a local optimum for the objective TT within 10 minutes only if the number of jobs is below 120 for the case with 16 machines. Refer to the figures in Appendix E.2.

Other interesting observations can be made when comparing typical objectives that aim at OTD. The experiment E.6 compares due date related objectives with each other in different scenarios. Minimizing L_{max} takes the shortest run times and minimizing TT shows the longest run times, whereas minimizing TU is in between both. However, no complexity reduction rule exists between the objectives TT and TU. The observation that minimizing TU takes less time than minimizing TT provides indication that solving problems with unit penalties are easier to solve than those with TT. See Figure 62; cf. Appendix E.6.

8.1.2 Number of Machines

The number of machines determines the size of a problem in a way that the algorithm's run time generally increases with an increasing number of machines. The problem apparently becomes harder to solve with an increasing number of alternatives resulting from more machines. Refer to the experiment E.1 that provides scheduling results and run times for the problem $Rm \mid p-batch, b < n, fmls \mid \cdot$ with various objectives; cf. Appendix E.1. It turns out that VND takes no more than 10 minutes to reach a local optimum for a typical scheduling problem with 16 machines when the number of jobs is below 120. Figure 54 also represents the experiment E.1's results; it shows the results of a series of model instances in which the smallest model counts 4 machines with 120 jobs and the biggest model contains 16 machines with 240 jobs.

In a similar way Li et al. (2008, 2009a) observe that more machines require longer computation time.

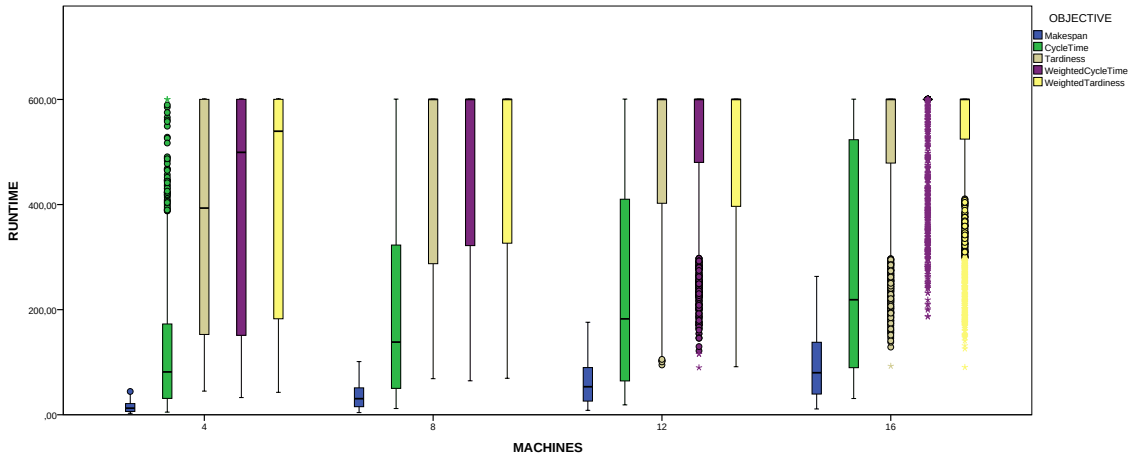


Figure 54: Run times for the objective functions depending on the number of machines (experiment E.1)

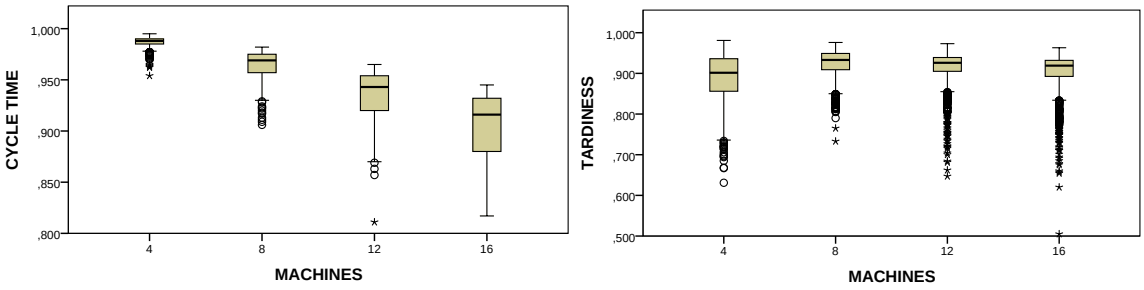
The experiments indicate that the number of machines has a relatively strong effect on the optimization results. The improvement in the objectives, i.e. TCT and TT, increases with an increasing number of machines as long as the maximum computation time limit is not exceeded. In relation the experiments indicate that the number of machines has a stronger effect on the optimization results than the number of jobs. There is another observation regarding the objective functions: the TCT objective shows a higher sensitivity to the number of machines than the TT objective. Refer to Figure 55 depicting the results related to the problem $Rm \mid p-batch, b < n, fmls \mid \cdot$ with the objectives $\sum C_j$ and $\sum T_j$; cf. Appendix E.2.

Similarly, Li et al. (2008, 2009a) evaluate a variant of a BPM scheduling problem, finding that more machines lead to a more significant improvement of the objective TWT. In the same line Tajan et al. (2012) show that the improvement in CT increases with decreasing number of processors.

The observation that a model with more machines leads to better results points to a certain advantage of the scheduling paradigm compared to dispatching: the partitioning effect. The classical dispatching approach does not look across machines and therefore fails in finding a proper mapping between machines in parallel and the jobs waiting in front of them. The results clearly indicate that a higher number of machines leads to greater improvements in the objective as long as the size of the problem remains manageable. As a result the conclusion can be drawn that scheduling a work center is more worthwhile when it contains more machines.

The next observations additionally support the point of view that setting reasonable computational deadlines for certain model instances is a key factor for deploying a scheduling system. In the case of exceeding computing times the improvement in the objective approaches zero, e.g. the improvement in TT is nearly zero when trying to find an optimized schedule for 240 jobs on 12 machines or more. Interestingly, the improvement dramatically shrinks despite of the fact that the

run time increases rapidly. This observation is a strong argument for the assumption stating that the search space becomes too large for being searched effectively. Consequently, it becomes obvious that increasing run times do not lead to better results in every case. But as long as the procedure is not aborted due to the maximum run time limit, i.e. the method identifies a local optimum, the improvements increase with increasing run time, respectively with increasing problem size. Refer to the figures in Appendix E.2.



(a) total cycle time

(b) total tardiness

Figure 55: Improvements depending on the number of machines (experiment E.2)

8.1.3 Number of Jobs

Besides the number of machines, the number of jobs determines the size of a problem. The problem becomes apparently harder to solve with an increasing number of alternatives resulting from more jobs, which generally cause an increase in run time. When comparing the factors machines and jobs it becomes obvious that the number of the jobs has a stronger effect on the run times than the number of the machines. This observation probably serves as an indication that sequencing is computational more expensive than partitioning.

It turns out that finding an optimum with VND for problems with 160 jobs and four machines takes less than 10 minutes in all cases, respectively for all objectives under study. The experiment E.1 is designed to evaluate the effect of the number of machines based on the problem $Rm \mid p - batch, b < n, fmls \mid \cdot$ with various objectives. Refer to Figure 56; cf. Appendix E.1.

In the same line Tajan et al. (2011) note that the computational time increases rapidly with an increasing number of jobs and Li et al. (2008, 2009a) also mention that more jobs require longer computation time.

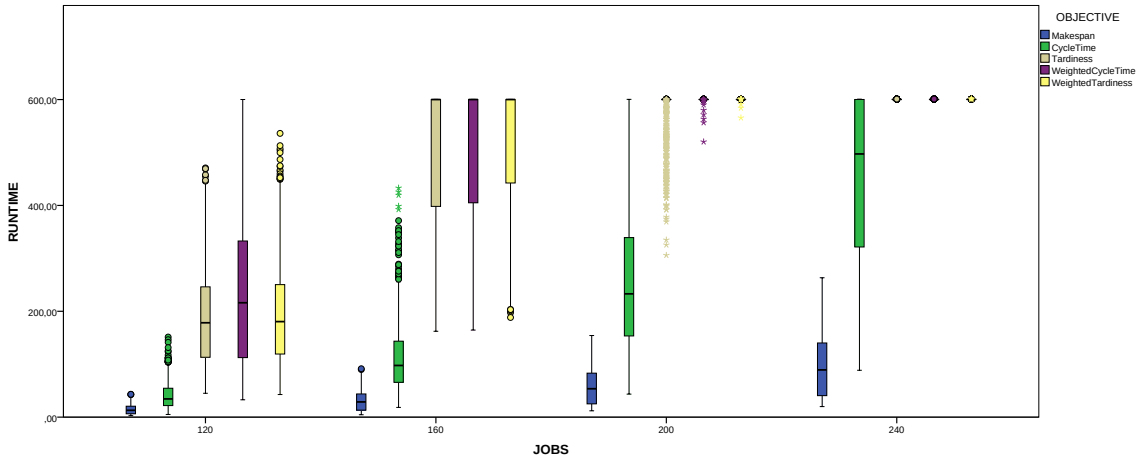


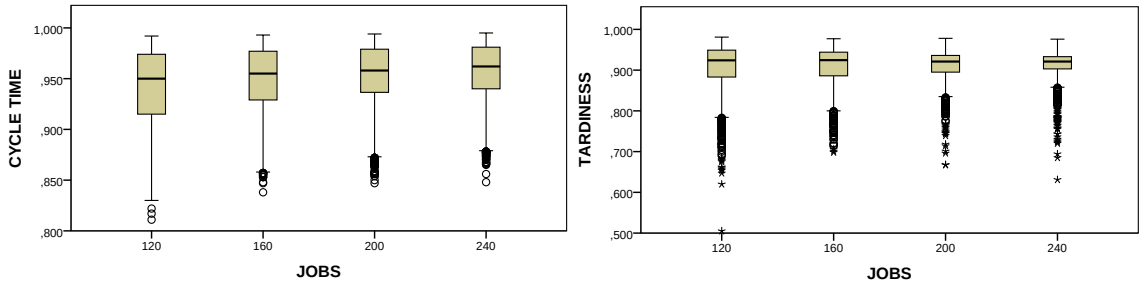
Figure 56: Run times for objective functions depending on the number of jobs (experiment E.1)

Another important observation reveals interesting differences regarding the objective functions

and their relationship to the number of the jobs. The improvement in TCT decreases with an increasing number of jobs and the best results emerge for the scenario with 16 machines and 120 jobs. In contrast to the TCT objective the improvement in minimizing TT increases with an increasing number of jobs as long as the computational deadline is not exceeded. In cases of exceeded computing times the improvement in the objective approaches zero, e.g. the improvement in TT is nearly zero when trying to find an optimized schedule for 240 jobs on 12 machines or more. But as long as the procedure is not aborted due to the maximum run time limit, i.e. the method identifies a local optimum, the improvements increase with increasing run time. This means the improvements respectively increase with an increasing number of jobs, which directly represents an increasing problem size.

The experiment E.2 investigates the problem $Rm \mid p-batch, b < n, fmls \mid \cdot$ with the objectives $\sum C_j$ and $\sum T_j$, providing run time measures that depend on the number of jobs. Refer to Figure 57; cf. Appendix E.2.

The complexity-increasing effect of the number of jobs is observed multiple times in literature. For example Dobson and Nambimadom (2001) experimentally show that the number of the jobs affect the performance of their solution schemes.



(a) total cycle time

(b) total tardiness

Figure 57: Improvements depending on the number of jobs (experiment E.2)

8.1.4 Number of Job Families

The constraint of incompatible job families is one of the main characteristics of the focused scheduling problem. It is interesting to know how the number of job families, in particular in relation to the number of machines, effects the search process and its results.

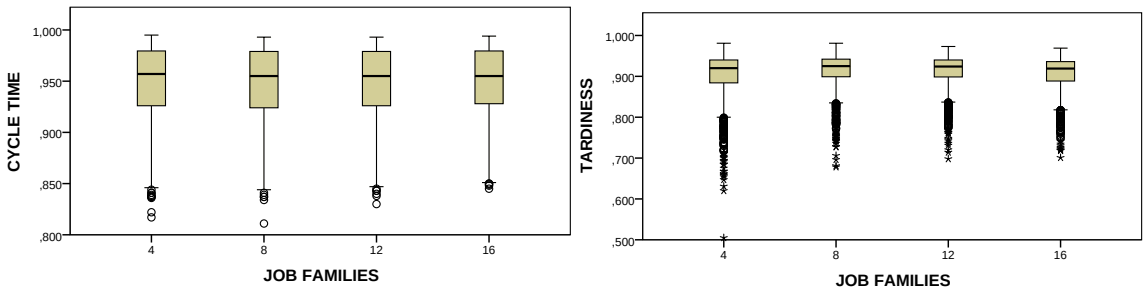
The experiments show that the search behavior clearly depends on the number of job families. Interestingly the behaviors observed in conjunction with changing numbers of job families also changes with the objective. The objectives TT and TCT are studied more in detail through the experiment E.2 in order to figure out their dependency on the number of job families. The experimental results provide indication that the job families show no significant effects in the case of the TCT objective; the improvement in CT seems to remain unaffected by the number of job families despite of the observation that the run time slightly increases with the number of job families. The experiment E.2 investigates the problem $Rm \mid p-batch, b < n, fmls \mid \cdot$ with the objectives $\sum C_j$ and $\sum T_j$ in order to examine the effect of the number of job families. Refer to Figure 58; cf. Appendix E.2.

Similar statements can be found in the related literature. Dobson and Nambimadom (2001) present experimental results showing that the number of families does not affect the performance of their solution schemes. But Tajan et al. (2008, 2012) note that the reduction in CT is typically larger with more job families.

In contrast to the TCT objective the TT objective depends considerably on the number of job families. It can be observed that the improvement in TT increases with the number of job families in the case of four machines. Interestingly this behavior turns around in the case of additional machines, e.g. for the case of 12 and 16 machines the TT improvement decreases with the number of job families. However in most cases the run times decrease with the number of job families when minimizing TT, which indicates that the number of alternatives decreases as well. Since Figure 58

shows no clear behavioral trend regarding the number of job families, refer to the diagrams in Appendix E.2.

The effect of the number of job families is discussed further in the context of batch sizes (Section 8.1.6), machine eligibility constraints (Section 8.1.7), and non-identical job sizes (Section 8.1.8).



(a) total cycle time

(b) total tardiness

Figure 58: Improvements depending on the number of job families (experiment E.2)

Especially the run time behavior depending on the number of job families changes with the objective function. For example the run time increases with an increasing number of job families when the objective is to minimize C_{max} or TCT. In contrast the run time decreases with an increasing number of job families when the objective is TT, TWT or TWCT. The experiment E.1 investigates the problem $Rm \mid p-batch, b < n, fmls \mid \cdot$ with various objectives. Refer to Figure 59; cf. Appendix E.1.

Similar observations derive from other research work. Yao et al. (2012) state that problems with a large number of job families are more difficult than those with a small number of job families. Their experiments show that solving problems with larger number of job families is more time-consuming compared to those with less job families. Li et al. (2008, 2009a) also show that more job families require longer computation time. On the contrary Azizoglu and Webster (2001) find that an increased number of job families makes problems easier to solve because there are fewer jobs per family and consequently fewer possible solutions, which makes the problem less computationally intense.

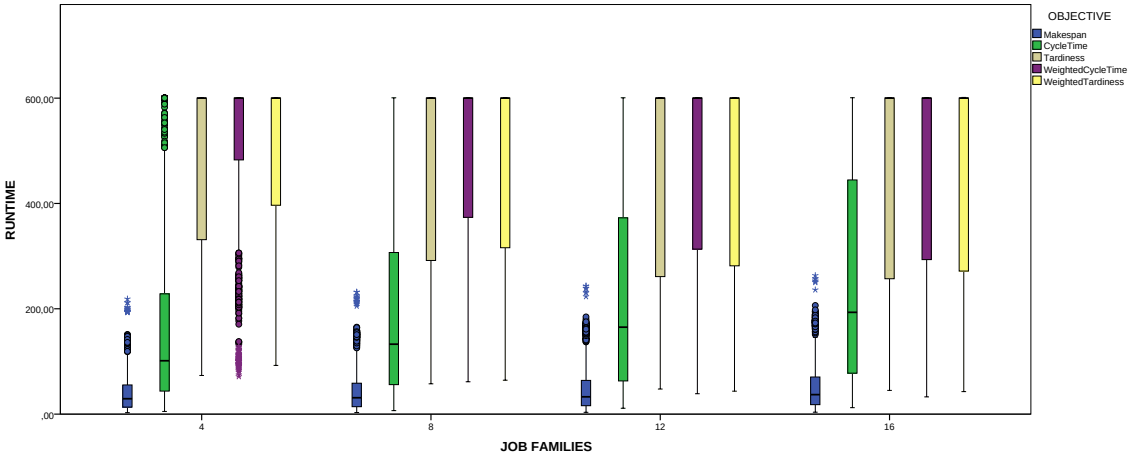
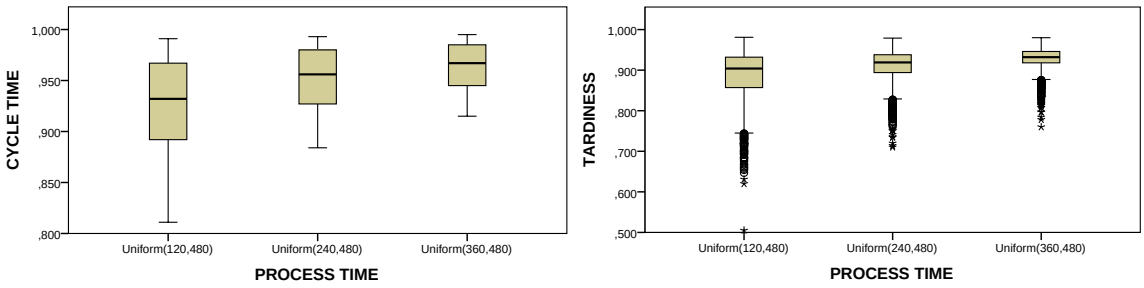


Figure 59: Run times for objective functions depending on the number of job families (experiment E.1)

8.1.5 Process Time

The processing time is naturally one of the most decisive factors. The variety in the processing times of the jobs is clearly a source of optimization potential, offering numerous alternatives to create a schedule. The experiments question the way the processing time has an impact on the search procedure; the influence on the objectives is of particular interest.

The relationship between the process time scheme and the improvement in TCT and TT is studied further in the experiment E.2. The experimental results suggest that the process time scheme influences the level of improvement in a way that a higher variance in the processing time causes greater improvements in TCT. The same effect, albeit weaker, is observable for the objective TT. The greatest improvements are gained for the cases with highly variant processing times, which seem to assist greater improvements by increasing the number alternatives in the scheduling decisions. The experiments clearly show that this effect is stronger for the objective TCT than for the objective TT. The experiment E.2 evaluates the effect of different process time schemes for the problem $Rm \mid p-batch, b<n, fmls \mid \cdot$ with the objectives $\sum C_j$ and $\sum T_j$. Refer to Figure 60; cf. Appendix E.2. In addition the experiment E.3 that deals with batch sizes in Section 8.1.6 provides results underpinning the observation that higher variances in processing times lead to greater improvements in the objectives; the experiment E.3 is based on the problem $Rm \mid p-batch, b<n, fmls \mid \cdot$ with the objectives $\sum C_j$ and $\sum T_j$.



(a) total cycle time

(b) total tardiness

Figure 60: Improvements depending on the process time scheme (experiment E.2)

The experiment E.1 shows that the run times remain unaffected by the processing time scheme in most cases, with one exception: the run times slightly decrease with decreasing processing time variance when minimizing TCT is the objective. The TCT objective particularly depends on the process time scheme, i.e. on the distribution of the processing times. Many different processing times caused by a wide range in the processing time distribution lead to many different possible combinations in the schedule, whereas a lower processing time variance decreases the number of alternatives. This thesis is underpinned by the results from the experiment E.1 designed to study the effect of the processing time in the case of the problem $Rm \mid p-batch, b<n, fmls \mid \cdot$ with various objectives. Refer to Figure 61; cf. Appendix E.1.

The specific role of processing times is a question in other research works. Sung et al. (2002) note that the computational burden on their proposed algorithm dramatically increases with the number of different processing times.

8.1.6 Batch Sizes

The experiment E.3 is designed to investigate the effect of the batch size on the search behavior. It is intended to clarify the relationship between the improvement of the objectives (TCT and TT) and the distribution of the batch sizes.

The experiment E.3 focuses on minimizing TCT or TT, comparing the results with SPT and EDD dispatching. It is designed to evaluate the effect of different batch sizes based on the problem $Rm \mid p-batch, b<n, fmls \mid \cdot$ with the objectives $\sum C_j$ TCT and $\sum T_j$ TT. The maximum batch size varies between two and eight, chosen from the array (2, 4, 6, 8). There are three batch size factors chosen from the array (0, 0.5, 1.0), defining different levels of uniform batch size

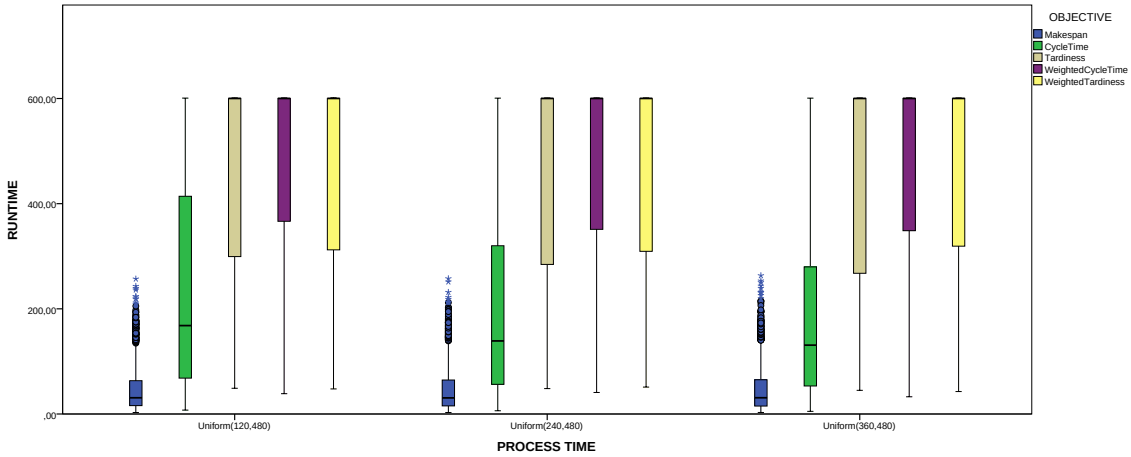


Figure 61: Run times for the objective functions depending on the process time scheme (experiment E.1)

distributions. Every design point is characterized by a maximum batch size value and a batch size factor. For example the design point with a maximum batch size of eight and a batch size factor of 0.5 corresponds to model instances with batch sizes that range from four to eight (distributed uniformly). A batch size factor that equals one leads to model instances where all batch sizes are set to the maximum batch size value, i.e. chosen from the array (2, 4, 6, 8). Using this experimental design, it is possible to investigate the effect of the absolute maximum batch size value as well as the distribution of the batch sizes.

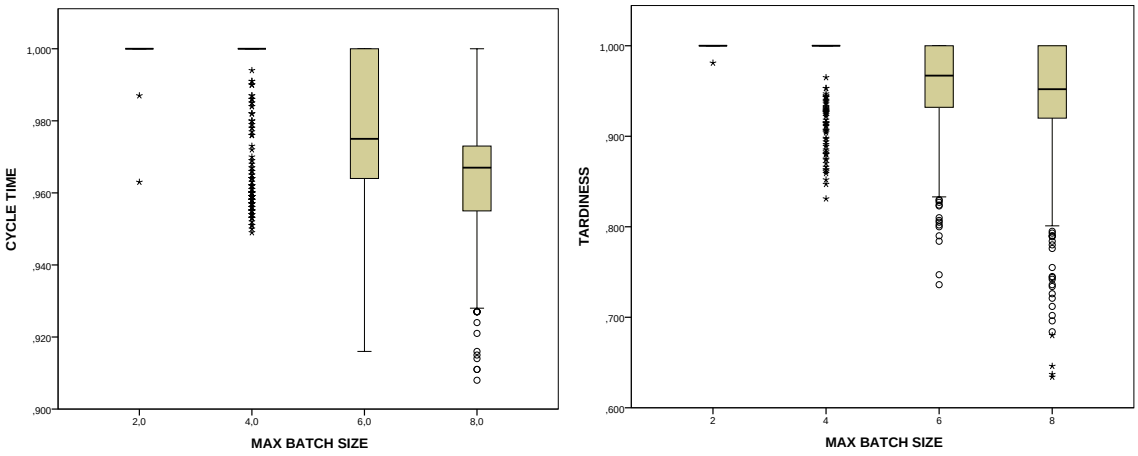
The experiment shows that the improvements both in TCT and TT strictly increase with increasing batch sizes. Adversely, the improvements decrease with smaller batch sizes. Furthermore, the improvement for the batch size factor 1.0 is higher than for factor 0.5, which indicates that a higher maximum batch size level leads to greater improvements. The biggest improvements can be observed for the case with a constant maximum batch size of eight. Interestingly there is no improvement in CT for the smallest batch size models despite of the fact that the run time is relatively high. Refer to Figure 62; cf. Appendix E.3.

The related literature documents similar observations. Li et al. (2008, 2009a) find that bigger machine capacity leads to a better improvement of the TWT. However, Lee and Uzsoy (1999) focus on C_{max} minimization and find that their heuristic methods' performance decreases with increasing machine capacity.

Despite of the fact the optimization procedures for TCT and TT display similar behavior with respect to the improvement in the objectives, there are differences in the run time behavior. The run time for the TT objective increases with increasing batch size whereas the run time for the TCT objective shows the greatest values for the smallest batch sizes. However, the improvement for the scenarios with the smallest batch sizes draw near zero. Consequently, the algorithm performs a considerable amount of search activities but does not gain considerable improvements compared to those models with greater batch sizes; cf. Appendix E.3.

The observations in the related literature can be seen as standing in contrast to the results from the experiment E.3. Sung and Choung (2000) and Sung et al. (2002) find that the method's running time increases with increasing batch sizes when C_{max} is minimized. Yao et al. (2012) examine a problem with CT objective, stating that the problems with a large batch size are consistently harder to solve than those with the small batch size. Li et al. (2008, 2009a) find that bigger machine capacity leads to lower computation time when TWT is minimized.

The run time for the TT objective increases with the number of job families for small batch sizes (2, 4) and seems to decrease with the number of job families for greater batch sizes (6, 8). The run time for the TCT objective increases with the number of job families for the smallest batch size models (2), but remains diffusively unaffected by the job families for greater batch sizes. Refer to the experiment E.3 for another view on the effect of the job families. The experiment also



(a) total cycle time

(b) total tardiness

Figure 62: Improvements depending on the batch size and the objective function (experiment E.2)

shows that the range in the processing times also influences the improvements: a higher variance in processing times leads to higher improvements in the objectives; cf. Appendix E.3.

8.1.7 Dedication Density

The experiment E.4 investigates machine eligibility constraints mirrored by a density factor in the models. It is designed to evaluate the effect of the density factor at the case of the problem $Rm \mid p - batch, b < n, fmls, M_j \mid \cdot$ with the objectives $\sum C_j$ and $\sum T_j$. Machine eligibility constraints define a set of machines that are allowed to process a certain job family. The mapping between a machine and a job family is also known as dedication, i.e. the process is dedicated to a machine. The density factor describes the ratio between the number of the actual dedications and all possible dedications. A density factor identical to one would mean that every job family is allowed to process on every machine, whereas a lower density factor implies that some job families are forbidden on certain machines.

The experiment E.4 focuses on the objectives TCT or TT with a density factor ranging from 0.1 to 1.0. In addition the number of job families is varied in order to study potential correlation between job families and machine eligibility constraints.

The experiment shows that the effect of the dedication density also depends on the objective. The observed behavior for the objective TCT differs from that documented for the objective TT. The average improvement in TCT increases with an rising density factor, which means that a uniform work center without any process dedications provides the best environment to minimize CT by optimization. In contrast the average improvement in TT decreases with an increasing density factor, which means that a higher number of process restrictions supports tardiness optimization. Refer to Figure 63; cf. Appendix E.4.

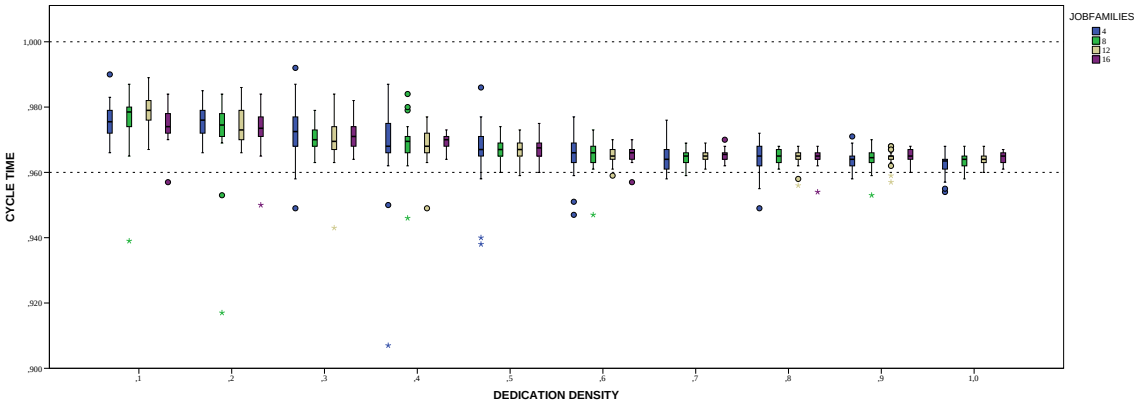
However, the spread in the improvement decreases for both objectives with increasing density factors. The variance for the objective TT is generally higher than that for the objective TCT, especially for low dedication factors. The reason for the varying spread in the improvements probably lies in the particular model instance, i.e. the specific structure of the mapping between machines and job families. The actual mapping between machines and job families, favorable or unfavorable, for those model instances with low density factors determines whether there is high or low improvement to be optimized. Consequently for some models there is not much to minimize while other models offer more optimization potential, even for the case with identical dedication factors.

For both objectives, TCT and TT, the run time increases with increasing density factor. This can be seen as an indication for the thesis that a higher density factor implies fewer forbidden machine-job combinations and therefore increases the number of alternatives that in turn increases

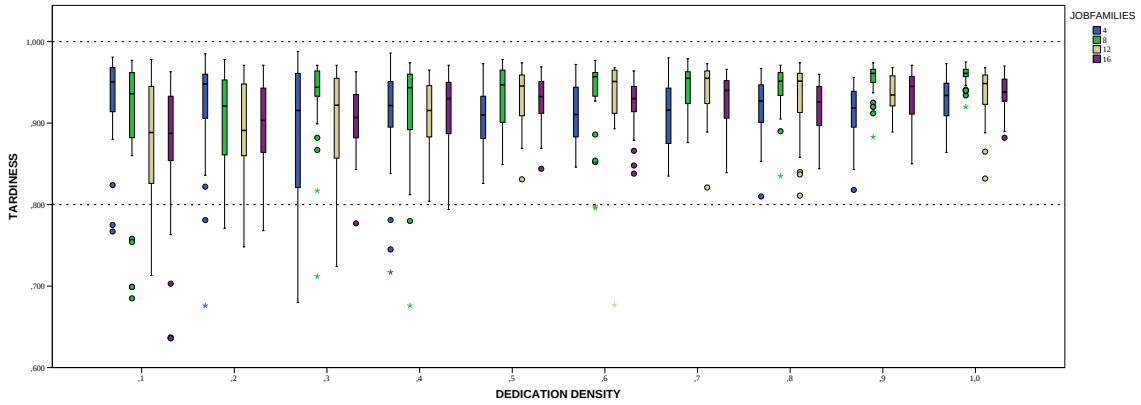
the complexity and the search method's run times. Refer to the run time diagrams in Appendix E.4.

The experiment E.4 also investigates the role of the number of job families in the context of machine eligibility constraints. Again the behavior between the objectives differs with respect to improvements and run times. The improvement in TCT seems to evolve independently from the number of job families, whereas the improvement in TT obviously depends on the job families. Refer to Figure 63; cf. Appendix E.4.

Despite of the observation that the TCT improvements show no dependency on the job family, the run times show an interesting pattern that indicates a linkage between search method and job families. On one hand the run times decrease with increasing job families for low dedication density factors and on the other hand the run times increase with increasing job families for high density factors. When minimizing TT the run time decreases with an increasing number of job families for most of the cases. Refer to the run time diagrams in Appendix E.4



(a) total cycle time



(b) total tardiness

Figure 63: Improvements depending on the dedication density (experiment E.4)

8.1.8 Job Sizes

The experiment E.5 was designed to investigate the effect of the job sizes. It is designed to evaluate the effect of the job sizes in the case of the problem $Rm \mid p - batch, b < n, fmls, M_j \mid \cdot$ with the objectives $\sum C_j$ (TCT) and $\sum T_j$ (TT). In the context of small lot size manufacturing policies it is interesting to know whether the job size plays a considerable role in a scheduling system.

The experiment E.5 covers eight different job size schemes accompanied by four different job family schemes. The size of the jobs is uniformly distributed between a minimum threshold chosen from the array $(3, 6, 9, \ldots, 24)$ and the maximum size of 25 wafers. The different variants are simply denoted in the diagrams with the minimum threshold value. Consequently there are two

extreme design points existing out of eight variants. On one hand the model instances in which the job size is uniformly distributed between 3 and 25 are denoted with the number 3 in the model. This model implies that many jobs with non-identical and small sizes need to be scheduled. On the other hand those model instances where the job size ranges between 24 and 25 wafers are denoted with 24 in the diagram. In this case no small lots exist in the model since all lots count either 24 or the maximum number of 25 wafers. The objective is either minimizing TT or TCT.

The search method behaves contrary for the focused objectives with respect to different job size models. With respect to the objective TCT the experiment shows that the improvement decreases with increasing job size. Put it another way, where more small lots need to be scheduled there can be more improvement in TCT expected by optimization. If the goal is TT minimization then the behavior turns around in a means that the improvement in TT increases with increasing job size in most cases. There is one exception when minimizing TT: if the number of job families equals four then the improvement decreases with increasing job size. This means that a high number of small lots does not positively contribute to the goal of minimizing TT of the scheduled jobs when the number of job families is above four in this scenario. Refer to Figure 64; cf. Appendix E.5. Similarly, Dobson and Nambimadom (2001) show that the size of the jobs affects the performance of their solution schemes.

Both the run times and the number of traced search moves increase with increasing job size for both objectives. But there is one exception in the run time behavior when minimizing TT: if the number of job families is identical to four the run time tends to decrease with increasing job size unless the minimum job size threshold is below 15. From that point onwards he run time tends to increase again. Refer to Appendix E.5.

In the related literature similar observations can be found. Azizoglu and Webster (2001) show that an increasing job size variation is associated with higher computing times. Dupont and Dhaenens-Flipo (2002) note that where there is a greater variety of job sizes there are more problems that are difficult to solve, given that the problem instance is not too small.

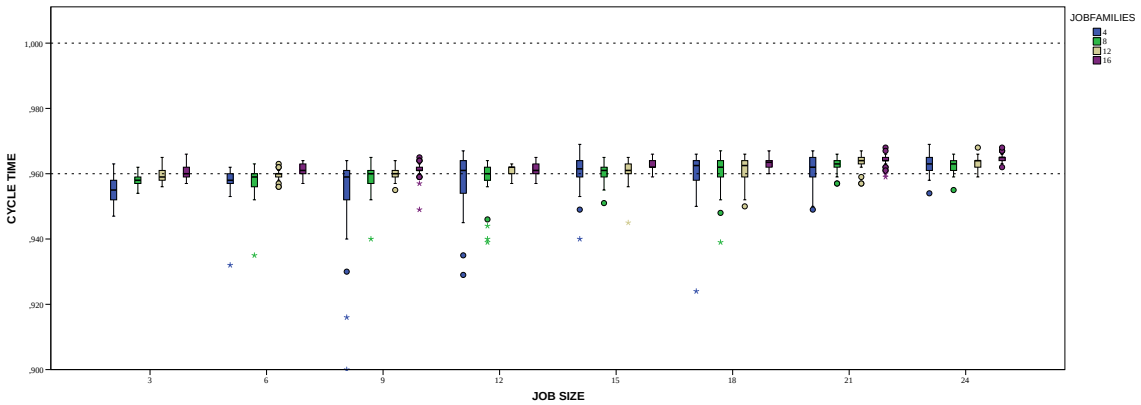
The experiment offers more results that are suitable to investigate the effect of the job families in the context of non-identical jobs sizes. The results indicate that the relationship between the improvement gained by optimization and the job families changes with different job size scenarios. Interestingly the search method's behavior depends on the objective, i.e. differs between both objectives. The improvement TCT decreases with an increasing number of job families. This effect is more obvious for small job size scenarios and less apparent for those job size scenarios with fewer small jobs. In contrast the TT improvement increases with the number of job families in most cases. The greatest improvements show up for the models with 16 job families and nearly no small jobs. There is an exception present for the lower half of the jobs size scenarios, respectively those models with more small lots. In those cases the improvement in TT is greater when the number of job families is four compared to the relating eight and twelve job family cases. This behavior directly corresponds to the measured run times: the longest run times are observed for the models with four job families accompanied with many small lots. Refer to Figure 64; cf. Appendix E.5

8.1.9 On-Time Delivery

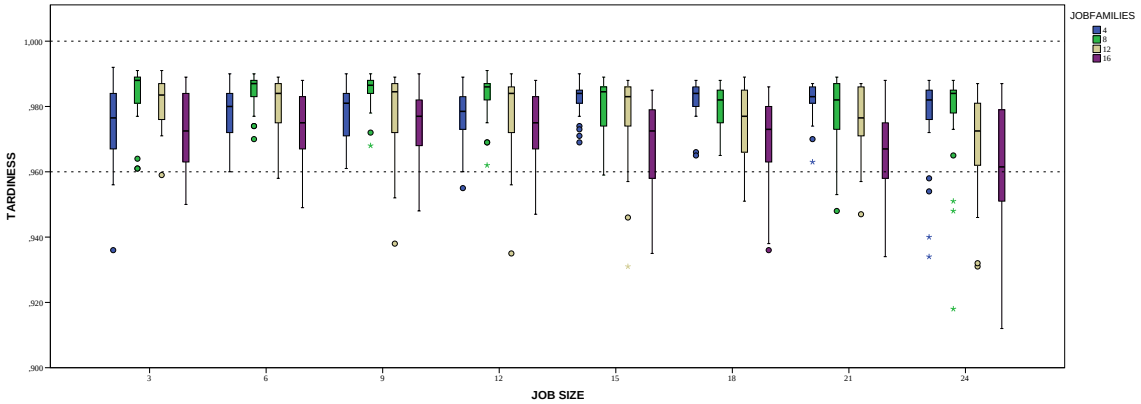
The experiment E.6 takes a closer look at due date related objectives; it employs the problem $Rm \mid p-batch, b < n, fmls \mid \cdot$. Especially the setting of the due dates, e.g. tight or loose, is under investigation. Furthermore it is taking a step into the direction of multi-objective optimization by figuring out how due date related objectives affect each other. For example the question is raised what happens with the L_{max} if the optimizer minimizes TT.

The experiment E.6 studies three objectives related to due dates: *a*) L_{max}, *b*) TT, and *c*) TU. But the central goal of this experiment is to examine the effect of the initial mean and variance settings of the due dates on the improvements in the objectives. The due dates are drawn from a normal distribution with varying mean and variance. A negative due date means that the job is already too late and a positive due date indicates that the job still has the opportunity to be scheduled in time. The due date mean value ranges from -48 to +48 hours in steps of 12 hours and the variance is either set to 12, 24 or 36 hours. The optimization results are compared to the EDD dispatching solution that simultaneously serves as the initial solution.

The experiment shows that the initial setting of the due dates affects the optimization results



(a) total cycle time



(b) total tardiness

Figure 64: Improvements depending on the job sizes (experiment E.5)

considerably. Defining the distribution from which the initial due dates are drawn during the model generation procedure implicitly determines the improvements gained by optimization in the aftermath.

It is highly desired to choose a favorable value range for setting the due dates in order to create a model characteristic that allows considerable improvements through optimization. In other words, unfavorable model settings exist for which no optimization method will be likely to generate acceptable results, which are suitable for discussion. The experiment shows for the objectives TT and TU that a due date mean value between zero and twelve hours delivers the best results. The best improvement results for the L_{max} objectives are observed for mean values of 36 and 48 hours. However, the improvement generally decreases in the area of the extremes, i.e. when the due dates are too tight or too loose. Common for all three objectives is that the model scenarios with the lowest variance in the initial tardiness show the highest improvements in all cases.

Further interesting observations can be made when comparing the objectives with others, in particular their effect on the other due date related performance measures. Minimizing L_{max} has nearly no effect on the other objectives; it only affects the L_{max}. Minimizing TT simultaneously leads to less TU and has nearly no effect on the L_{max} with one exception: L_{max} slightly increases for scenarios with very tight due dates. Minimizing TU adversely affects the distribution of both the TT values and the L_{max}. Refer to Figure 65 and the diagrams in Appendix E.6.

The run times basically evolve synchronously with the improvements; they decrease with an increasing absolute due date mean value. It becomes obvious that minimizing TT and TU takes more time than minimizing L_{max}. This underpins the fact that minimizing L_{max} is the easiest problem between these three with respect to the findings from the complexity theory. The experiment shows that the initial due date setting influences their run times and hereby gives indication when it is easier to minimize TT or TU.

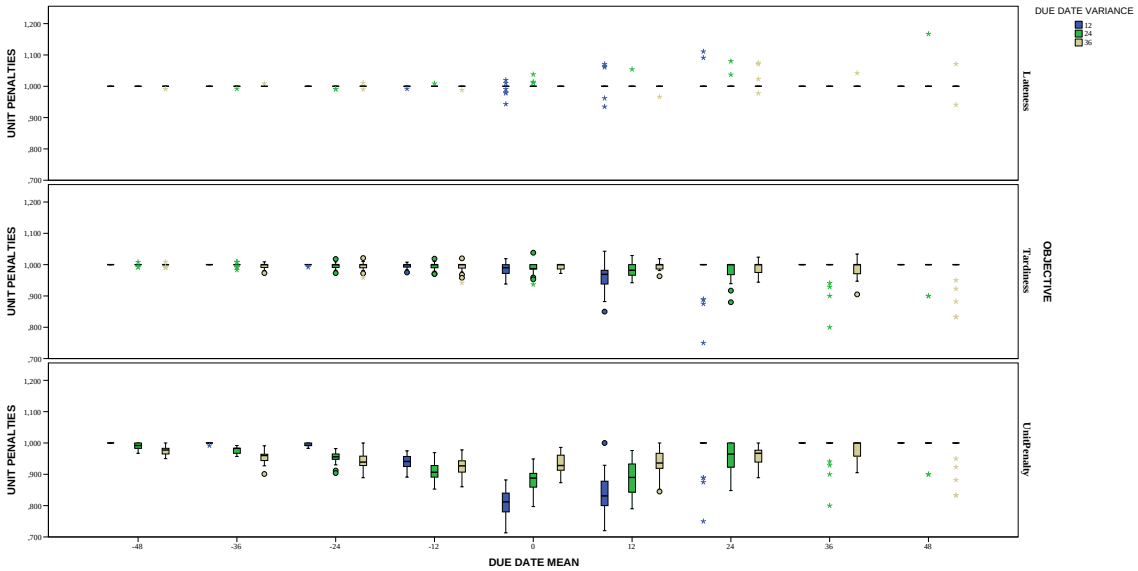


Figure 65: Improvements depending on the initial due date setting and the objectives (experiment E.6)

8.1.10 Priorities

Job priorities play a very important role in the scheduling of waferfabs. Usually the goal is to minimize TWCT and/or TWT, taking into account that some high-priority jobs are more important than others. Consequently it becomes necessary to incorporate job weights when minimizing the objectives.

The experiment E.7 is designed to investigate the search behavior with respect to different job priorities. It is based on the problem $Rm \mid p - batch, b < n, fmls \mid \cdot$ with the objectives $\sum w_j C_j$ (TWCT) and $\sum w_j T_j$ (TWT). Initially the question is raised whether the search behavior and its

results change with different numbers of priority classes. It is interesting to know if there is a situation, i.e. with few or many priority groups, where the effect of optimization is improved or impaired. Furthermore the experiment intends to clarify whether the value of the weighting factor has an influence.

The experiment E.7 covers model instances with different numbers of priority classes ranging from 2 to 10, chosen from the array (2, 4, 6, 8, 10). In addition the priority weight factor is varied in the same way, ranging from two to ten in steps of two. The objective is to minimize TWCT or TWT.

The experiment shows that the improvement in TWCT and TWT increases with a rising number of priority groups, albeit slightly. On the contrary the priority weight factor shows no clear pattern of influence. See Figure 66 and the diagrams in Appendix E.7. The frequency diagram in Figure 67 clearly shows that the resulting solution describes a schedule in which the high-priority jobs are processed first.

The run time generally increases with an increasing number of priority classes; cf. Appendix E.7.

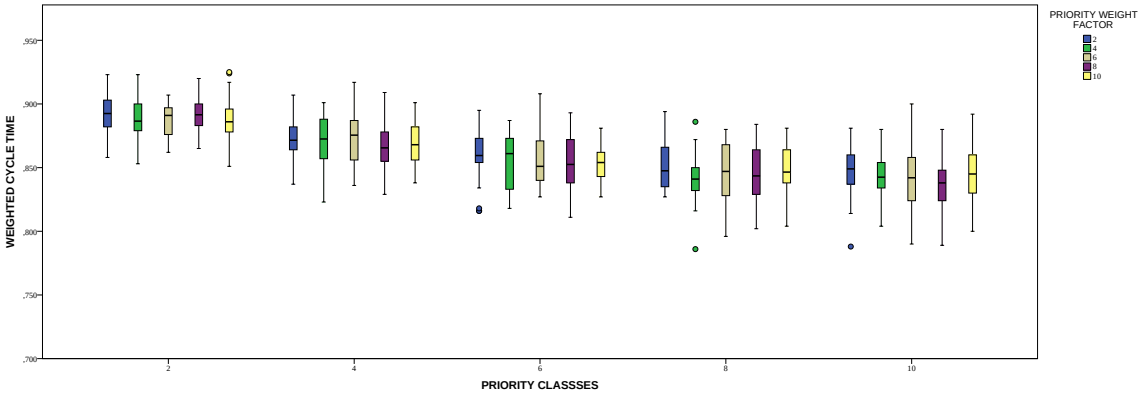


Figure 66: cycle time improvements depending on the number of the job priority settings (experiment E.7)

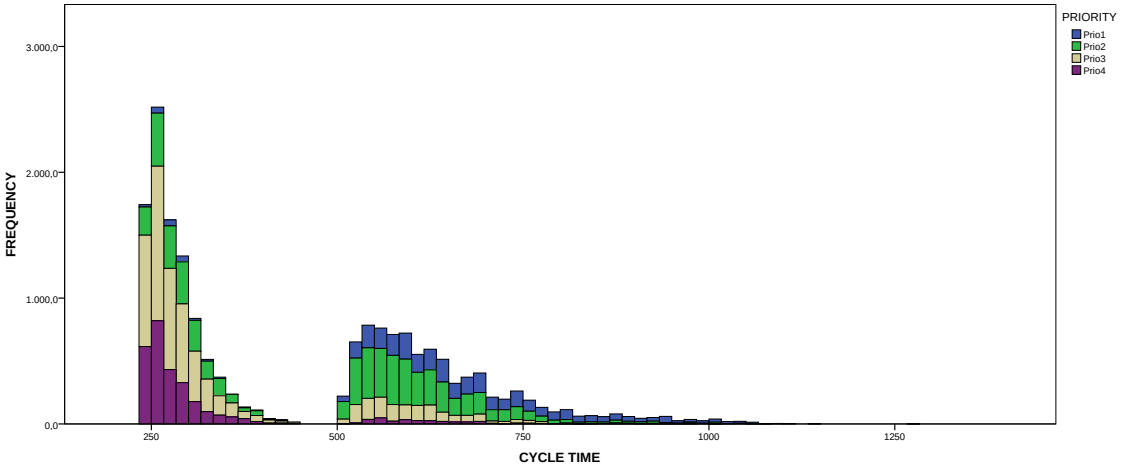


Figure 67: Frequency diagram for the cycle time with priority classes (experiment E.7)

8.1.11 Correlation Between Objectives

The experiment E.8 takes a closer look at the correlation factors between the objectives at hand of the problem $Rm \mid p - batch, b < n, fmls \mid \cdot$ with several objectives. The performance measures are mutually interconnected; some of them are closely interlinked with each other, while others

are not. It is common knowledge that those relationships exist, but very little is known about the dimensions of these relationships. The experiment E.8 is designed to analyze the correlations between a selection of important measures/objectives in order to develop a better understanding.

The objectives under study are: *a*) C_{max}, *b*) TCT, *c*) TWCT, *d*) L_{max}, *e*) TT, *f*) TWT, *g*) TU, and *h*) TWU. The idea behind this is to optimize for a specific objective, and then to calculate the correlation factors through Pearson's method between the focused objective and the remaining measures. The resulting matrix of correlation factors provides exact values that indicate the direction and the strength of the inter dependencies between any combination of two measures.

According to Pearson it is possible to derive a few observations from the calculated matrix of correlation factors that represent the strongest interconnections, given in decreasing order of their size of impact. Figure 68 shows the experimental results. There are four main observations: *a*) The basic objectives correlate with their weighted versions (0.8+). *b*) OTD measures correlate, but L_{max} is exceptional (0.7+). *c*) The TCT and the C_{max} share a loose relationship (0.5-0.6). *d*) The TCT slightly improves OTD measures (0.3-0.4).

a) The objectives correlate with their weighted versions. the basic objectives such as TCT and TT are naturally strongly connected with their weighted counterparts TWCT and TT respectively. TWT and TT show high values around 0.9 for their correlation factors against each other, in both directions. Likewise it is possible to observe correlation factors above 0.7 between TWCT and TCT.

b) The OTD measures generally correlate, but the L_{max} is somehow exceptional (0.7+). The due date related objectives TT and TU share a strong connection with each other, characterized with a correlation factor above 0.7. The corresponding weighted variants TWU and TWT also positively affect the unweighted counterparts TT and TU. Interestingly the objective L_{max} is widely isolated within the set of the due date related objectives.

c) The TCT and the C_{max} share a loose relationship (0.5-0.6). The objectives related to THP positively affect CT measures. Minimizing C_{max} results into lower TCT, indicated by a correlation value around 0.6. It is possible to observe a slightly lower impact on TWCT, where the correlation factor is approximately 0.5.

d) The CT objectives slightly improve OTD measures (0.3-0.4). Minimizing TCT has a minimizing effect on due date related objectives, e.g. TU (0.3), TWU (0.3), TT (0.4), and TWT (0.3).

8.1.12 Pareto Objectives

In practice it is most often the intention to minimize multiple objectives. The experiment E.9 is designed to analyze bi-criteria objective functions, which combine two criteria equivalent to each other (pareto scheme). The underlying problem can be described with $Rm \mid p-batch, b < n, fmls \mid \cdot$. The resulting bi-criteria objective function considers solutions as improved if both measures show improvements, accepting that one of them remains unchanged. Nevertheless, the problem arises that different objectives can contradict each other. For example Koehler and Khuller (2013) state that there are instances where minimizing TCT and C_{max} simultaneously is impossible. Rose (2002) notes that minimizing CTs and maximizing OTD are conflicting goals.

This experiment considers 10 different bi-criteria objective functions in pareto fashion, each composed of two components out of *a*) C_{max}, *b*) TCT, *c*) TT, *d*) L_{max}, and *e*) TU. Consequently, the following 10 pareto scenarios are examined: *a*) C_{max} and L_{max}, *b*) C_{max} and TT, *c*) C_{max} and TU, *d*) TCT and C_{max}, *e*) TCT and L_{max}, *f*) TCT and TT, *g*) TCT and TU, *h*) TT and L_{max}, *i*) TT and TU, and *j*) TU and L_{max}. The models are optimized in three directions for each scenario, i.e. in the directions of the two basic objectives and in the direction defined by the corresponding pareto function. Every optimization run uses schedules obtained by a random dispatching rules as the initial solution and for the reference in order to maintain fair starting points for all objective functions. The results shown in the figures (Appendix E.9) are given in relation to the initial solution.

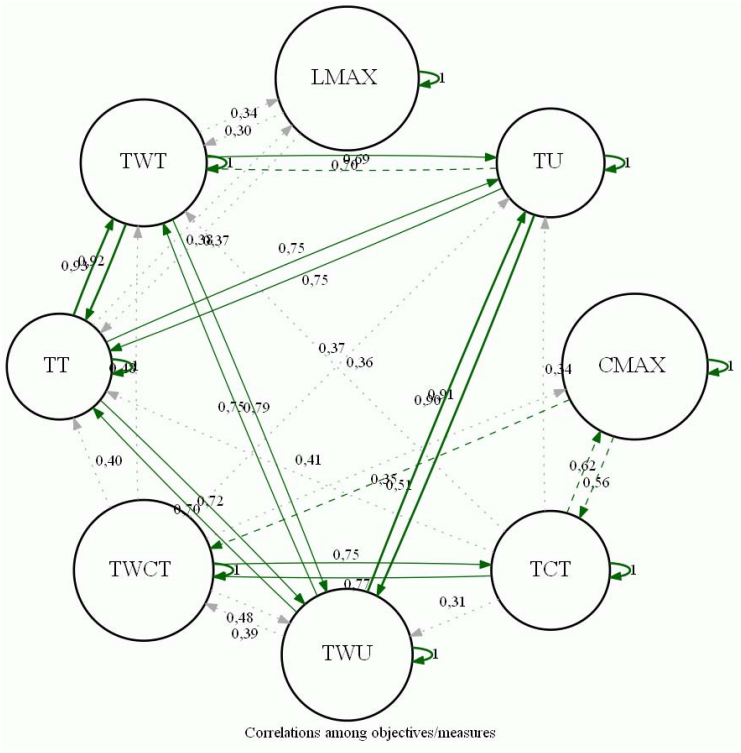


Figure 68: Correlation factors between objectives (experiment E.8)

The experiments show that two the objectives either agree or disagree, whereas some combinations behave neutrally. The combined pareto objective often orientates on one of both objectives and some pareto objectives nearly behave identical to a single objective. For example minimizing C_{max} and TT as single objectives can lead to contradictory results, whereas simultaneously minimizing both in pareto fashion performs well. Figure 69 shows that minimizing C_{max} may deteriorate TT and minimizing TT clearly increases C_{max} in most cases. The pareto objective combines both very well and even discovers new optima in C_{max}. Refer to Appendix E.9 for visualizations of the 10 pareto scenarios.

8.2 Dynamic Models

Incorporating future job arrivals into the decision making process remarkably empowers optimizing schedules for scheduling problems with BPMs, e.g. in a waferfab's furnace area. Especially the decision when to start a batch is important in two respects. On hand it is beneficial to delay a batch in order to wait for one or more lots arriving soon. On the other hand when having the knowledge that no further lots are to arrive soon, it is better to start a smaller batch immediately. Taking a decision in either of the both directions requires the knowledge of future job arrivals predicted for a certain time horizon.

If no knowledge about future job arrivals exists the decision whether to start a job is also referred to as batch loading policy. For example the MBS policy is a simple and popular BPM control policy. The problem with the MBS policy is to determine the optimal or near-optimal batch sizes, bearing in mind that optimal batch sizes change with the level of utilization. Determining proper values for the batch sizes particularly becomes problematic in a multi-product fabrication characterized by incompatible job families.

So called look-ahead strategies overcome the problem of finding suitable batch sizes by including upstream information about the next arriving jobs into the decision process. This section experimentally examines the use of job arrivals as one of the most important aspects in a batch scheduling system.

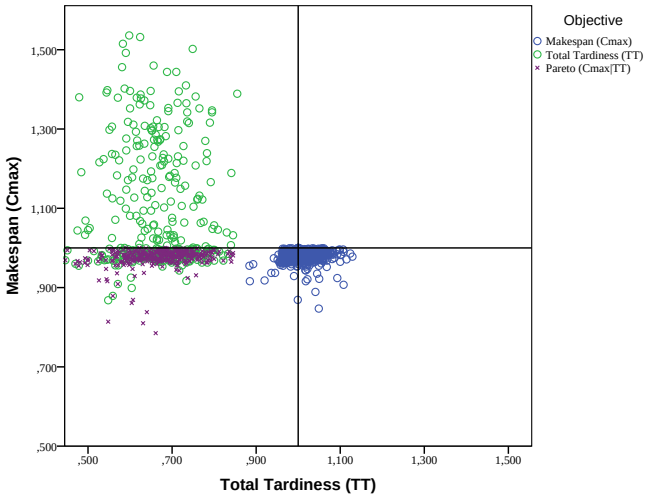


Figure 69: Minimizing makespan and total tardiness as single objectives and in pareto fashion (experiment E.9)

The Default Settings for Model and Method The basic model comprises 8 machines and 480 jobs to schedule. The utilization level varies between 0.7, 0.8 and 0.9 and the arrivals are uniformly distributed from zero to the expected C_{max}. 30 independent model instances in each design point ensure that the results are considered as representative and resistant to single model effects.

The scheduling procedure is accompanied by a TWD scheme; cf. Section 8.2.1 for corresponding experiments. The TWD interval equals 10 minutes which means that the TWD procedure creates a new scheduling problem every 10 minutes. The VND search starts with a FIFO schedule for each sub problem and minimizes TCT within maximal 30 seconds per time frame. The method setting defines six neighborhoods: *a*) split a batch, *b*) merge two batches, *c*) swap two batches, *d*) move a batch, *e*) swap two jobs, and *f*) move a job. See Table 23 for the default settings.

The Experiments The discussions in this section refer to six experiments used to evaluate important factors in the case of problem instances with dynamic job arrivals. The experiments E.10 and E.11 investigate the TWD scheme, respectively the effect of the width of the time window. The experiments E.12, E.13 and E.14 take a closer look at the look-ahead horizon. Since no prediction is free of errors the experiment E.15 determines the effect of errors in job arrival predictions. The experiments E.10 to E.15 deal exclusively with the problem $Rm \mid p-batch, b<n, r_j, fmls \mid \sum C_j$. Refer to Table 24 for an overview on the experiments carried out.

8.2.1 Time Window Decomposition (TWD)

TWD is an essential decomposition technique used to disassemble scheduling problems on the timeline into smaller sub problems, each sequentially and separately solved by an optimization procedure. Hereby it is possible to evaluate scheduling methods for large problem instances with practical relevance.

The experiments E.10 and E.11 examine the effect of TWD, more precisely the effect of the width of a time window on the TCT. It turns out that the optimization results rely on the time window size. The experiment E.10 ranges the time window value from 10 to 90 minutes in steps of 10 minutes. The experiment E.11 provides results for a tighter time window value grid where the value is incrementally increased from 10 to 90 minutes.

One might assume that the smallest time window, considered synonymously to an event based simulation like DES, would lead to the best results in any case. But this is not unconditionally true, as proven by the experimental results. It is true so far as the TCT tends to increase with an increasing time window size. The following linkage between time window size and TCT can be seen: the larger the time window the lower is the probability to start a certain job or batch just at

Table 23: Default settings for the series of experiments related to the dynamic model

factor		level
basic model parameter	machines	8
	jobs	480
	utilization	0.7, 0.8, 0.9
machine parameter	job families	8
	dedication density	$1(uniform)$
	processing times	$U(240, 480)$
	batch size (lots)	8
	batch size (wafers)	200
job parameter	job sizes	$s_j = 25$
	job priorities	-
	job due dates	-
	job arrivals	$r_j \in U(0, Cmax)$
	job arrival errors	-
model instances per parameter combination		30
method parameter	initial solution	FIFO
	objective(s)	TCT
	VNS type	VND
	deadline	30 sec
	time window interval	10
	look-ahead horizon	90

Table 24: Overview of the experiments related to the dynamic model

experiment	studied factors	model instances	total runs
E.10	utilization, interval	90	810
E.11	utilization, interval	90	7290
E.12	utilization, look-ahead	90	720
E.13	utilization, look-ahead	90	810
E.14	utilization, look-ahead	90	7290
E.15	utilization, look-ahead, arrival errors	3240	19440

the moment of its arrival. Given that no look-ahead information is used the scheduling procedure is limited to those jobs that arrived before the current time window begins. Consequently the jobs that arrive during the current time window cannot be scheduled before the next time window begins. Refer to Figure 70; cf. Appendix E.10 and the model-specific diagrams in Appendix E.11 in particular.

Applying decomposition techniques such as TWD is linked to uncertainties with regard to the optimization potential. TWD can have strong effects on the results, which can change with the interval width in particular. There are values for the time window size that lead to considerably better results than the smallest possible time unit does. This is an interesting result. Since the scheduling system deals with discrete events, e.g. discrete job arrivals over time, it is reasonable that varying time window sizes can lead to varying results. The results show that the smallest time window does not necessarily lead to the best result with respect to the focused objective. The time window size in conjunction with randomly distributed job arrivals strongly effects the results. It can be deduced from the experiments that decreasing intervals tend to lead to better results on average, but not necessarily for a single model instance. Consequently it is not necessary to choose the smallest time window that comes with long run times in order to achieve the best results. It is often sufficient to choose an acceptable small time window, which enables good results within acceptable run times. Refer to Figure 70; cf. Appendix E.10 and the model-specific diagrams in Appendix E.11 in particular.

It is also remarkable that applying an optimization technique in conjunction with TWD does not always lead to improvements. The experiments show the phenomenon that some values for the time window interval lead to solutions outperformed by their corresponding dispatching reference. In fact the pure dispatching solution performs better than optimization combined with TWD. This is of special interest because the pure dispatching solution (without TWD) is obtained by the identical dispatching rule used to create the initial solution for the optimization procedure.

The following paragraph intends to give a reasonable explanation for this phenomenon. TWD disassembles the given scheduling problem, creating a sequence of sub problems solved and optimized sequentially. Each scheduling decision taken within a time window (sub problem) has an effect on the next sub problem in sequence. Every decision once taken remains effective for all succeeding sub problems. The key point is that an optimized solution for a single sub problem may cause unfavorable situations (compared to the non-optimized dispatching solution) leading to a sequence of sub problems where the optimization procedure does not compensate for the early scheduling decisions. As a result the final solution is not improved, but corrupted by the optimization activities compared to the pure dispatching solution. The improvements realized by applying VND vary between -10% and +20%, although the overall improvement is positive. Small changes in decomposition intervals may lead to significant changes in the results.

The crux of the matter is that a lack of information makes finding the optimal solution nearly impossible. The observed interrelationship between job arrivals and interval length points to the discrete nature of the entire system as well as to a structural weakness of scheduling approaches using TWD. Proper studies designed to generate reliable statements on the optimization potentials need to frame a sufficiently high number of independent model instances evaluated with varying time window sizes when TWD is applied. Especially for industrial applications (or cost-benefit calculations via simulation) it is hard to gather an adequate number of real-world models from history in order to evaluate the discussed scheduling methods to be introduced to production systems. Refer to Figure 70; cf. Appendix E.10 and the model-specific diagrams in Appendix E.11 in particular.

Obviously the run times depend considerably on the time window interval, respectively its width. The total run time increases with decreasing time window interval. Consequently the lowest total run time occurs for the widest time interval. The reason is quite simple: the smaller the time window the more time windows arise and subsequently more optimization problems need to be solved. Furthermore the run time increases with the utilization level since a higher utilization level corresponds to a higher number of jobs per time window, which in turn increases the size and the complexity of the problem. However the number of moves does not depends on the time window width but only on the utilization level. Refer to Figure 71; cf. Appendix E.10 and Appendix E.11.

Another observation stems from the fact that an increasing utilization level implies that the number of jobs to be scheduled per time window increases. That is the reason why the run times

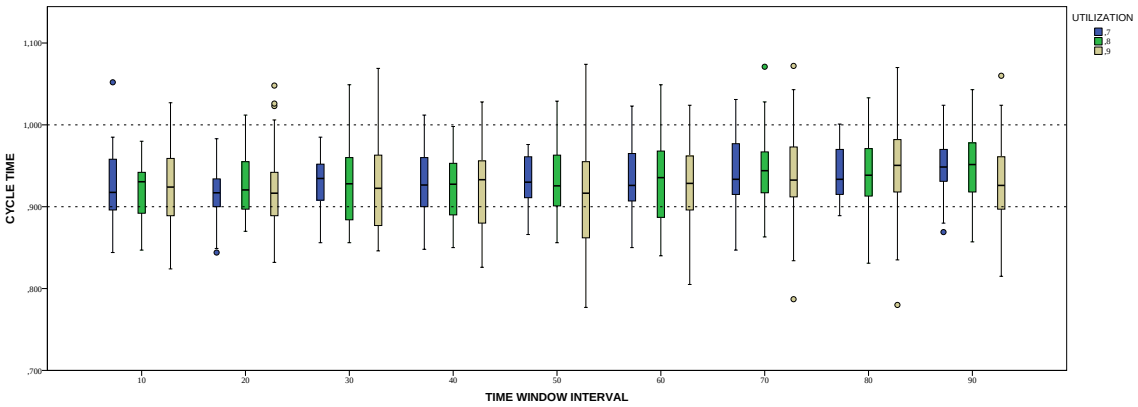


Figure 70: Cycle time improvements depending on the time window interval (experiment E.10)

as well as the number of traced search moves also increase with the utilization level. Refer to Appendix E.10 and the model-specific diagrams in Appendix E.11 in particular.

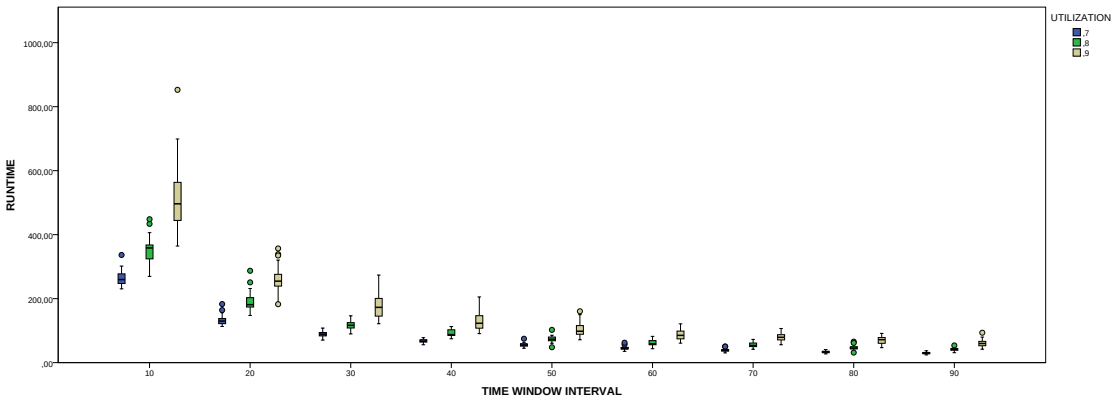


Figure 71: Run times depending on the time window interval (experiment E.10)

8.2.2 Look-Ahead Horizon

The term look-ahead stands synonymously for the scheduling's or control system's capability to incorporate predicted information about future job arrivals into the decision process. The look-ahead horizon defines the maximal time span to which expected job arrivals can be taken into account.

The experiments E.12, E.13 and E.14 show that the look-ahead horizon strongly effects the scheduling results, respectively the improvements gained by optimization. The experiments were carried out by applying TWD with a time window interval of 10 minutes. The process time is uniformly distributed between 240 and 480 minutes.

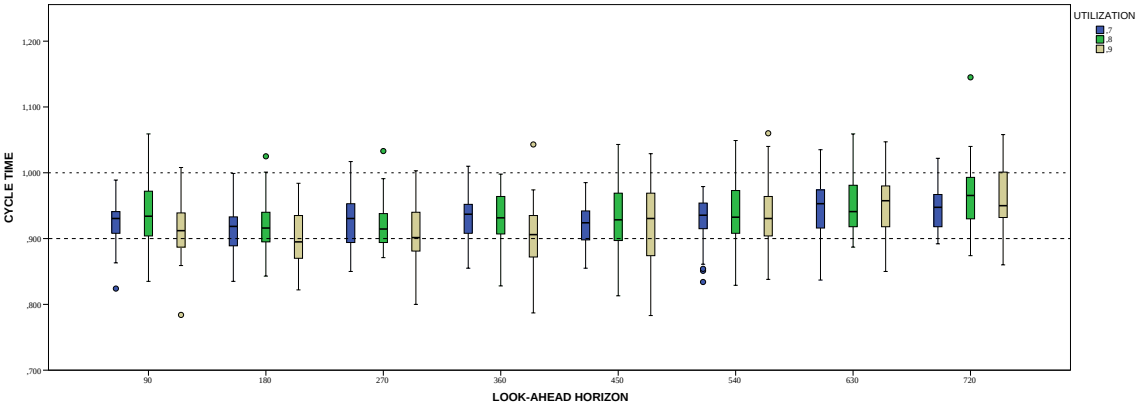
The results show that information about arriving jobs empowers the scheduling method remarkably. Figure 72 depicts a clear trend towards increasing improvements accompanied by an increasing look-ahead horizon for the small scale horizons .

The results show that the improvement gained by VND compared to FIFO dispatching tends to steadily increase with an increasing look-ahead horizon below 450 minutes. For look-ahead horizon values greater than 450 minutes the improvement degrades. This improvement decrease is most likely due to the fact that the problem instances become too large; the time deadline of 30 seconds is exceeded, triggering the optimization procedure to stop searching before a local optimum can be found. The observed improvements in TCT stay below 20% with a mean value below 10%. Refer to Figure 72.

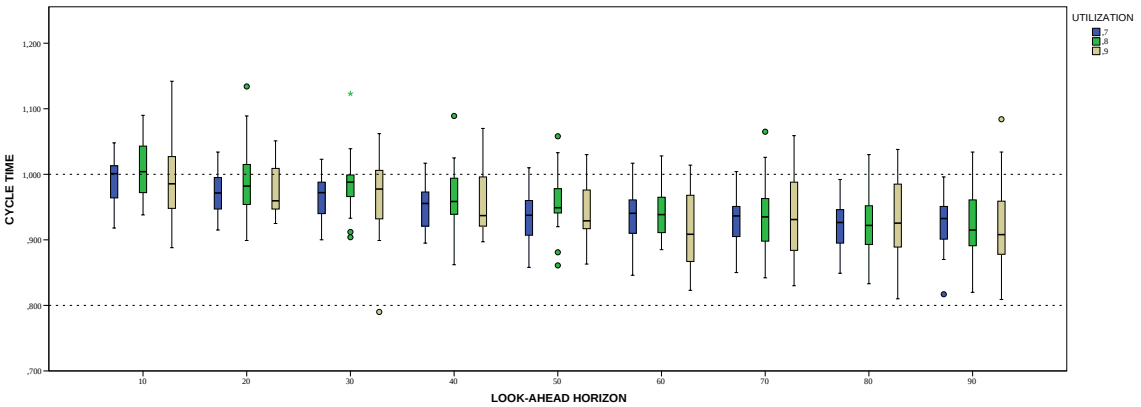
It is remarkable that the results also show negative improvements around -10% for individual look-ahead horizon values. A negative improvement mirrors a case in which the pure dispatching solution has a better objective value than the solution obtained by the search procedure. The reason is that the search procedure in conjunction with TWD and look-ahead information iteratively generates partial schedules with improved objectives. Unfortunately the partial schedules that are finally connected result in a schedule with impaired objective value compared to the pure dispatching solution. Refer to Section 8.2.1 discussing the same effect for the TWD procedure.

The frequency of optimized schedules that perform worse than pure dispatching (without look-ahead) is higher for short look-ahead horizons. Therefore it is recommendable to choose a proper look-ahead horizon that is long enough. Otherwise the improvement in TCT would draw towards zero despite the use of an optimization method. Refer to Figure 72; cf. Appendix E.12, Appendix E.13 and Appendix E.14.

The run time increases with increasing look-ahead horizon since the scheduling problem increases with the number of upcoming jobs in that time horizon. The limit of 30 seconds computing time per time window is reached for the look-ahead horizon of 630 minutes. From this point on the algorithm fails to discover a local optimum and is forced to abort by returning the current solution.



(a) big scale horizons (experiment E.12)



(b) small scale horizons (experiment E.13)

Figure 72: Cycle time improvements depending on the look-ahead horizon

8.2.3 Arrival Errors

Equivalent to the related dispatching approaches the batch scheduling strategies benefit extensively from look-ahead information, which provides information on predicted job arrivals over a certain time horizon. Unfortunately any prediction, regardless of the scope or intention, comes with errors

—and so does job arrival prediction.

Following the three laws of forecasting (Hopp and Spearman, 2001) any prediction comes with errors due to the fact that models used for prediction are always simplified abstractions. The lack of modeled system behavior, considered as negligible model details for the sake of model simplicity, unevitably results in prediction errors. Motivated by the fact that look-ahead information contributes remarkably to optimization potentials and by the fact that predictions are generally erroneous, the impact of look-ahead prediction errors on solution quality is of particular interest.

Usually studies discussing BPM scheduling problems with dynamic arrivals take those arrivals as fail-safe in their models. Due to the fact that look-ahead information is widely considered to be the largest source of optimization potentials in BPM scheduling problems and taking into account that any prediction is erroneous, the impact of prediction errors on scheduling benefits is of particular interest for practitioners. This section evaluates the impact of job arrival prediction errors on the solution quality, respectively their influence on optimization potentials.

The TWD scheme is extended in a way that predicted arrivals may be subject to disturbances in order to evaluate the effect of errors in job arrival predictions. Therefore the implemented model contains two dates for every job arrival: *a*) the predicted arrival date and *b*) the disrupted date that is finally used for schedule evaluation. The arrival dates are exposed to a normal distribution of offset errors, which may add a positive or negative delay to the originally predicted arrival. The prediction error depends on the actual look-ahead horizon for the predicted arrival. The prediction error for a certain job is calculated as the product of the time of the predicted arrival and a factor taken from a normal distribution. This calculation takes the fact into account that the further a predicted event lies in the future, the worse the prediction will be in terms of the accuracy. According to TWD any new time window is considered as a new scheduling problem. The VND search procedure improves the schedule by taking the predicted job arrivals without errors into consideration. Then the improved schedule is simulated again in order to incorporate the erroneous arrival dates.

The experiment E.15 defines 108 model types that vary in the utilization level, the arrival error scheme and the look-ahead horizon. The utilization level is set to 0.7, 0.8 or 0.9. There are 36 arrival error schemes, where 6 levels for the mean value and 6 levels for standard deviation value have been examined. The mean error varies between zero and 0.5 of the average processing time, respectively chosen from the array (0.0, 0.1, 0.2, 0.3, 0.4, 0.5). The value for the standard deviation also lies between zero and 0.5 of the average processing time, where the value belongs to the array (0.0, 0.1, 0.2, 0.3, 0.4, 0.5). Additionally the look-ahead horizon varies from zero to 90 minutes, chosen from (15, 30, 45, 60, 75, 90). For each model type 30 independent instances are created, which result in 3240 problem instances in total. The entire experiment takes 19440 optimization runs without those runs that provide the reference solutions. It provides noteworthy results that are used to discuss the effect of prediction errors on optimization potential.

When analyzing the experimental results the following observations can be made. First in accordance with previous studies the experiment shows greater improvements for higher look-ahead horizons, confirming the results in the experiments E.12, E.13 and E.14. Higher mean error values result in slightly lower improvements as expected. Interestingly the decrease in the improvement caused by arrival errors is very low. The effect of the standard deviation in the error distribution shows no clear behavioral trend in the visualized results. It is concluded that the level of variability has both positive (too early arrival) and negative (too late arrival) effects that neutralize each other. Furthermore the diagram shows that the level of utilization generates no difference in the improvement, neither for short look-ahead horizons nor for long look-ahead horizons. Refer to Appendix E.15; cf. Figure 73

Similarly Tajan et al. (2011) still observe significant improvements in TCT when the predicted arrival times are erroneous. It is commonly accepted that look-ahead control strategies are not very sensitive to errors in job arrival predictions, implying that job arrival predictions do not need to be very accurate to be useful (Glassey and Weng, 1991; Robinson et al., 2000; van der Zee, 2007). However it is clear that improving quality of predictions has a positive effect on the scheduling system.

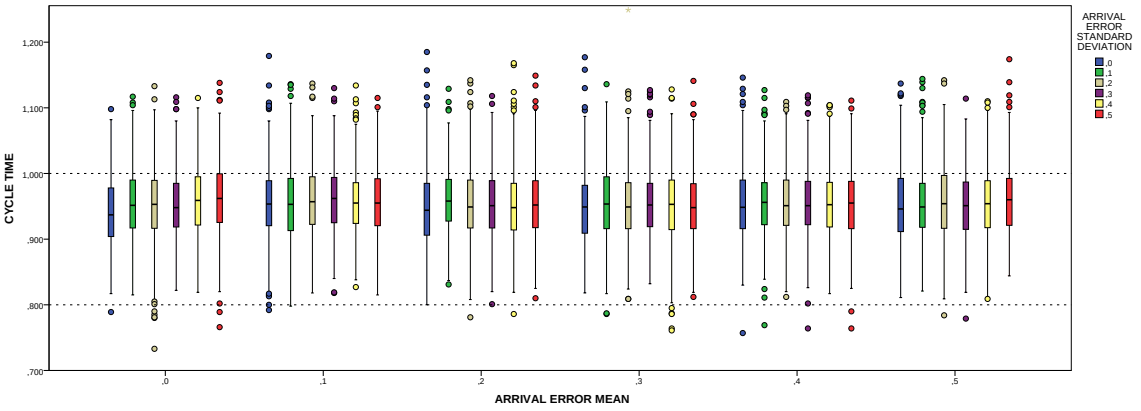


Figure 73: Cycle time improvements depending on the errors in job arrival prediction (experiment E.15)

8.3 Method and Benchmarks

This section investigates VNS more in detail, examining various method settings and their effect on the performance. The goal is to figure out which search method components contribute most effectively to the observed improvements, respectively which method setting performs best.

The Benchmark Models and The Default Method Settings The experiments investigated in this section make use of 90 model instances selected from a set of benchmark models described in (Doleschal, 2010). The selected benchmark instances describe the problem to schedule 80 jobs on 3, 4, or 5 parallel BPMs. The model instances differ in their characteristics, e.g. describe different utilization scenarios. The model instances formally belong to the class of problems denoted with $Rm \mid p-batch, b < n, r_j, fmls, M_j \mid \sum w_j T_j$. The experiments have in common that the initial solution is generated via BATC dispatching and the objective is TWT. The dispatching interval is set to five minutes, which means that every five minutes the dispatching procedure is executed, which probably results in a new job/batch started. The computational deadline is set to three minutes. The experiments are designed to study the five common VNS schemes: *a*) VND, *b*) RVNS, *c*) BVNS, *d*) GVNS, and *e*) VNDS. The search is performed through the use of the six neighborhoods: *a*) split a batch, *b*) merge two batches, *c*) swap two batches, *d*) move a batch, *e*) swap two jobs, and *f*) move a job.

This sequence of neighborhoods defines the default order of the neighborhoods iteratively visited during the search procedure. Since the search performs stochastically in most settings the experiment determines five replications per run in order to sufficiently cover stochastic effects on the analysis. The optimization results are given in relation to the initial solution (BATC) and in relation to the best CPLEX solution (with maximum run time of three minutes) taken from the benchmark data.

See Table 25 for a brief description of the benchmark models.

The Experiments This section comprises 12 experiments that take a closer look at the factors that influence the search performance of VNS. The first experiment E.16 investigates the role of the initial solution. The experiments E.17, E.18, E.19, and E.20 deal with the role of the neighborhoods as essential components in the VNS concept. Based on the results from the experiments E.21, E.22, and E.23 the LS phase is put into focus, whereby experiment E.22 takes a closer look at VNS schemes. The experiments E.21 and E.23 investigate LS with focus on different neighborhoods. The experiment E.24 provides results that help to discuss the shaking phase and the experiment E.25 and E.26 examine varying deadlines. The experiment E.27 is meant to document the search procedure in order to provide some measures suitable to characterize the search in a dynamic context.

Refer to Table 26 for an overview on the experiments carried out.

Table 25: Description of the benchmark models

factor	level	count
machines	3, 4, 5	3
jobs	80	1
utilization	0.7, 0.8, 0.9	3
job families	4	1
dedication density	machine specific dedication schemes (app. 0.7)	1
processing times	2(20%), 4(20%), 10(30%), 16(20%), 20(10%)	1
batch size (lots)	$B1 = 3, B2 = 4, B3 = 6, B4 = 4, B5 = 2$	1
batch size (wafers)	$B1 = 75, B2 = 100, B3 = 150, B4 = 100, B5 = 50$	1
job sizes	$s_j = 25$	1
job priorities	0.3(75%), 0.6(20%), 1.0(5%)	1
job due dates	$d_j \in r_j + U(0, Cmax)$	1
job arrivals	$r_j \in U(0, Cmax)$	1
problem parameter combinations		9
model instances per parameter combination		10
total number of model instances		90

Table 26: Overview of the experiments related to the benchmark studies

experiment	studied factors	method settings	total runs
E.16	initial solution, VNS schemes	75	28349
E.17	neighborhoods, VNS schemes	108	48600
E.18	neighborhoods, VNS schemes	27	12150
E.19	neighborhoods, VNS schemes	12	5400
E.20	neighborhoods	720	64800
E.21	local search, neighborhoods	18	1620
E.22	LS, VNS schemes	10	4500
E.23	LS, neighborhoods	6	2700
E.24	shaking range	18	8100
E.25	computational deadline	57	25650
E.26	VNS schemes, computational deadline	15	67500
E.27	GVNS search trace	2	18000

8.3.1 Initial Solution

The initial solution is generally one of the decisive factors in any metaheuristic optimization procedure. The experiment E.16 is designed to investigate the effect of the initial solution generated with dispatching according to the BATC rule further. Several authors discuss variants of BATC; cf. Section 6.5.2. The BATC rule requires the determination of a control parameter k. The question is raised which BATC factor leads to the best solutions and how strong the effect of the initial BATC solution on the final result is. The control parameter k is also known as the look-ahead factor.

The experiment E.16 investigates the performance of five variants of VNS, i.e. VND, RVNS, BVNS, GVNS, VNDS. The initial solution is generated with the BATC rule using a k value chosen from the array $(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.5, 2, 2.5, 3)$. A variant of BATC with a k value identical to 1.5 is denoted with BATC-1.5 in the diagrams. The experiment additionally comprises a more sophisticated BATC dispatching variant with varying k value, notated with BATC-VK. It evaluates five different solutions generated with k values from the array $(0.3, 0.4, 0.5, 0.6, 0.7)$ and returns the best.

Mehta and Uzsoy (1998) use k values ranging from 0.5 to 4.0 in their experiments. The BATC rule in the experiment E.16 performs best when the k value is between 0.4 and 0.6. However the differences between the results are rather minor. When comparing the results generated with VNS optimization, it becomes obvious that the VNS schemes that use BATC-VK lead to the best results. Comparing the single k factor variants, the methods for different initial solutions seem to perform nearly identical. But there is a noteworthy behavior: the results with the k value between 0.4 and 0.6 (beside BATC-VK) are characterized with less impaired results compared to the other variants. Refer to Figure 74; cf. Appendix E.16

The results imply that the VNS scheme is more important than the initial BATC solution. GVNS leads to the best results directly followed by VNDS, BVNS and VND in that order; RVNS shows the worst results. Refer to the diagrams in Appendix E.16.

Hansen and Mladenović (2003) note that the performance of VNS depends very little on the initial solution. Their observation is based on a VNS implementation for the minimum sum-of-squares clustering problem.

VND settles at a local optimum within 60 seconds in most cases. RVNS, BVNS, GVNS, VNDS perform stochastically, continuously trying to improve the solution until the maximum deadline of three minutes is reached. Interestingly the stochastic search schemes (RVNS, BVNS, GVNS, VNDS) perform different numbers of search moves until they are all terminated after running for three minutes. BVNS shows the highest number of moves performed within three minutes, closely followed by VNDS and GVNS. Then RVNS shows a considerably lower number of moves and VND clearly achieves the minimum number of moves. Refer to the run time diagrams in Appendix E.16.

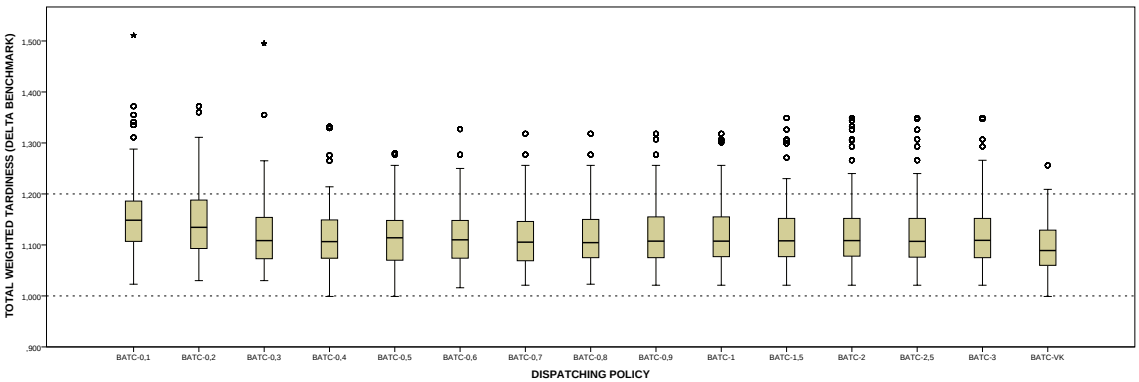


Figure 74: Total weighted tardiness improvement depending on the initial BATC solution (experiment E.16)

8.3.2 Neighborhoods

The definition of neighborhoods is a central element within the VNS concept. Although the basic VNS concept is relatively simple the different VNS schemes offer numerous possibilities to combine the neighborhoods in many ways. It is of particular interest how to include the neighborhood implementations in a VNS scheme. The question is raised whether a favorable neighborhood method setting exists.

Hansen and Mladenović (2003) emphasize the importance of designing adequate neighborhoods customized for the present problem. They recommend to continue spending a considerable effort on systematic studies of the best distributions of the neighborhoods, finding points of references that help to properly setup the different search levels and search phases; cf. (Resende and Ribeiro, 2009).

The implementation of the VNS concept comprises two search levels where each search level consists of a LS phase and a shaking phase. Both search levels may operate on different sets of neighborhoods. The different VNS schemes mentioned in the literature (e.g. BVNS, GVNS, and VNDS) describe different strategies of using the two search levels, trying to properly combine LS with shaking actions. BVNS, GVNS, and VNDS differ from each other in the way they use the two search levels. BVNS performs LS and shaking on the first search level and neglects the second level. GVNS performs the shaking phase operating on the first search level whereas the LS phase operates with the neighborhoods on the second level. VNDS performs LS and shaking on both levels. Section 7.2 describes the framework's VNS implementations more in detail.

The experiments E.17, E.18, E.19 and E.20 are designed to evaluate the contribution of the neighborhood implementations on the optimization process. The framework under study comprises six neighborhood implementations: *a*) split a batch, *b*) merge two batches, *c*) swap two batches, *d*) move a batch, *e*) swap two jobs, and *f*) move a job.

The experimental design covers VNS variants with different neighborhoods and/or sets of them assigned to the two search levels as part of the VNS schemes VND, BVNS, GVNS, and/or VNDS. The experiment E.17 investigates the performance of single neighborhoods, covering 36 different variants in which each search level contains a single neighborhood. One neighborhood out of the six is chosen for both, the first search level and second search level. The experiments E.18 and E.19 evaluate various combinations of neighborhoods, whereas experiment E.19 additionally puts focus on the neighborhood sequence. The experiments E.17, E.18 and E.19 frame the VNS schemes BVNS, GVNS, and VNDS for every of the mentioned combinations of neighborhoods. The experiment E.20 takes a closer look at the role of the sequence of the neighborhoods, respectively the order the neighborhoods that are visited during the search procedure. The VND is used as a search scheme in this case.

The results of the experiment E.17 show that some neighborhood combinations perform better than others. The best results are achieved with the VNS methods that move jobs, swap jobs, or merge batches. On the contrary splitting, moving, and swapping batches leads to less favorable results. This observation is underpinned by the search movements traced for the neighborhoods and their combinations. Splitting, moving and swapping batches accounts for considerably fewer search steps than the job-based neighborhoods operating with single jobs, i.e. move/swap jobs and merge batches. The results show that merging batches takes the largest amount of time and leads to the greatest improvements besides moving jobs from one batch to another. Considering all neighborhood combinations it turns out that no VNS scheme clearly outperforms another. It depends on the neighborhood combination if one VNS scheme performs better than another. Refer to the diagrams in Appendix E.17.

The experiment E.18 evaluates the combinations of three pairs of two neighborhoods. Better results occur for neighborhood combinations involving the two pairs split/merge batches and move/swap jobs. Between those the best results are displayed if the second search level operates on the job level, i.e. uses job swapping and job moving. VNDS often performs equally or better than GVNS, which in turn outperforms BVNS in most cases. Refer to Figure 75; cf. Appendix E.18. The experiment E.21 investigates neighborhoods in the light of different LS schemes. See Section 8.3.3 and Appendix E.21. In contrast Cakici et al. (2013) claim that repositioning the batches instead of jobs yields better results.

The experiment E.19 shows that the neighborhood sequence has nearly no effect on the

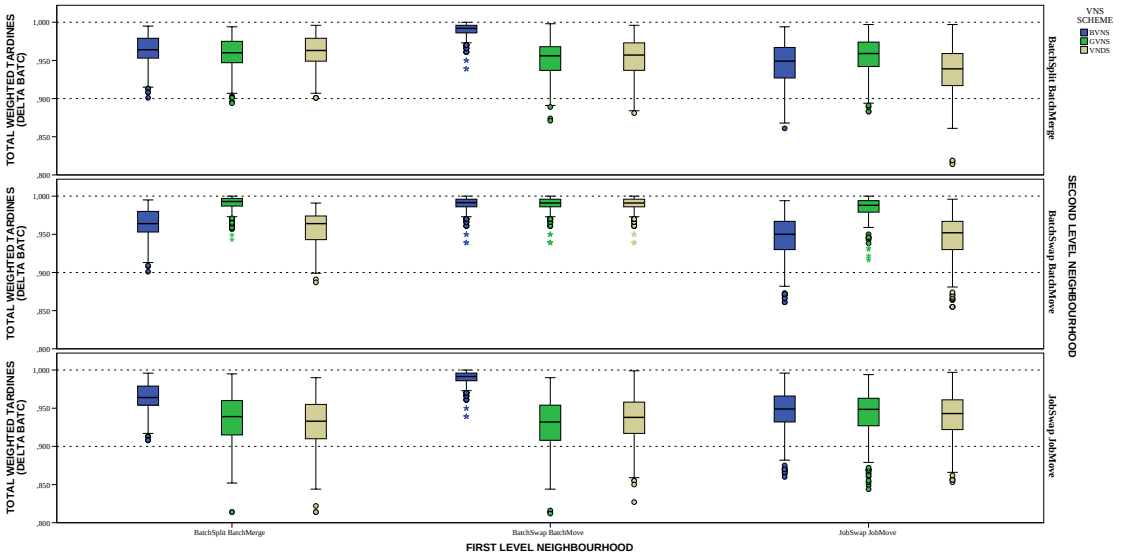


Figure 75: Total weighted tardiness Improvement depending on the neighborhood structure (experiment E.18)

performance of the VNS procedures. There is a very slight advantage for those method settings that start their search in the second level with moving and swapping jobs. Refer to the diagrams in Appendix E.19.

The experiment E.20 tests the VND scheme with 720 different neighborhood sequences, which simply represent all possible ordered sequences of the six neighborhoods. The difference between all these combinations of neighborhood sequences shows a maximum of 1% improvement. The run time results show that better improvements most often correspond to a higher number of search moves. Refer to Figure 76 and the diagrams in Appendix E.20. The experiment E.23 investigates neighborhood sequences in the light of different LS schemes. See Section 8.3.3 and Appendix E.23.

Zäpfel et al. (2010) state that the neighborhoods are typically ordered by increasing size of the neighborhood space. Hu and Raidl (2006) argue it is often critical to find a well-performing order of neighborhoods. They present a variant of VND search that dynamically changes the order of neighborhoods during the search procedure, presenting computational results indicating that their implementation requires substantially less time for finding solutions of comparable quality.

8.3.3 Local Search (LS)

Generally there are two basic LS schemes: *a*) best improvement search and *b*) first improvement search. The first improvement search policy moves to the next solution whenever a new improved solution is found. The best improvement LS procedure exhaustively explores the area around the current solution; first evaluating all possible solutions around the incumbent solution and then choosing the best solution to move on with. Section 5.5 describes the LS concept and typical issues related to it more in detail. The following experiments are designed to clarify which LS policy performs better under which settings.

The experiments E.21, E.22, and E.23 put focus on the LS phase. The experiment E.21 investigates LS in the light of neighborhoods and the experiment E.22 takes a closer look at different VNS variants. The experiment E.23 additionally examines the sequence of neighborhoods with respect to the LS policies. The experiment E.21 and E.23 make use of the VND search scheme, whereas E.22 compares different VNS schemes in conjunction with different LS policies.

The results of the experiment E.21 show no distinct difference in the solution quality between the two LS policies, i.e. between best improvement search and first improvement search. But the experiment does show that three neighborhoods outperform the remaining, i.e. merge two batches, swap two jobs and move a job. These three best-performing neighborhoods also show the highest run times since they perform considerably more moves to the optimum until VND

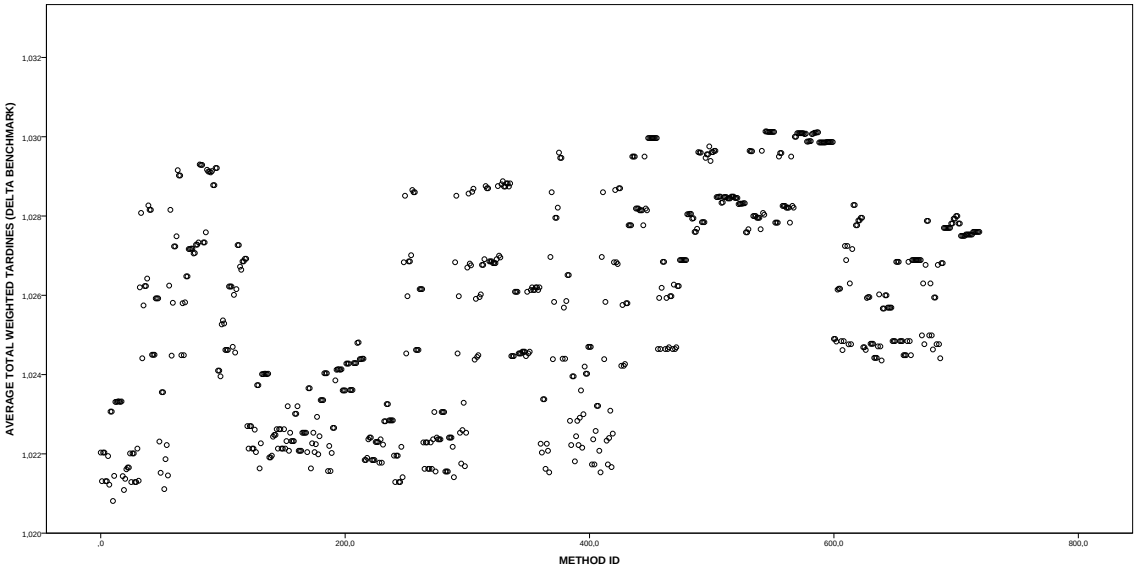


Figure 76: Total weighted tardiness improvement depending on the neighborhood sequence (experiment E.20)

terminates. This result is presented in accordance with the results in Section 8.3.2 examining the role of the neighborhoods in particular. When comparing the run times it becomes obvious that the best-improvement procedure needs more time than the first-improvement method. However their performance in terms of solution quality is nearly identical. Refer to Figure 77; cf. Appendix E.21.

The results of the experiment E.22 confirm the results of the experiment E.21 despite of the fact that the experiment E.21 uses VND and the experiment E.22 investigates BVNS, GVNS, and VNDS. The experiment E.22 also shows no clear difference in the performance between the two LS schemes under study, i.e. between best-improvement search and first-improvement search. The only difference is the number of moves the methods perform until the computational deadline is reached. The first-improvement LS scheme performs more moves than the best-improvement LS scheme, which indicates that the first-improvement LS scheme is capable of exploring more search space. However the first-improvement strategy does not outperforms the best-improvement strategy in terms of solution quality. Refer to the diagrams in Appendix E.22.

The third experiment examines the role of a LS policy in conjunction with two different neighborhood sequences. The first sequence is the default sequence used for all other experiments: *1*) split a batch, *2*) merge two batches, *3*) swap two batches, *4*) move a batch, *5*) swap two jobs, and *6*) move a job. The second sequence is exactly the reverse order. The experiment shows that the default neighborhood sequence slightly outperforms the reverse, but the LS has no effect neither for the one nor for the other sequence. Refer to the diagrams in Appendix E.23 as well as in Appendix E.19 and Appendix E.20; cf. Section 8.3.2.

Based on the experiments it is concluded that the first-improvement LS scheme seems to lead to the same results in less time compared to best-improvement LS. Similarly Resende and Ribeiro (2009) observe that both search strategies lead to the same solution while the first-improving strategy requires less computing time. They also state that it is more likely that the best-improving strategy converges earlier to a bad local minimum.

8.3.4 Shaking

The shaking procedure enables VNS to escape from local optima. The default shaking procedure describes one single random operation performed on the current solution, which is then used to start a LS procedure again.

The experiment E.24 is designed to find out if multiple shaking moves instead of one single shaking move can improve the VNS procedure. Employing the geographical terminology in

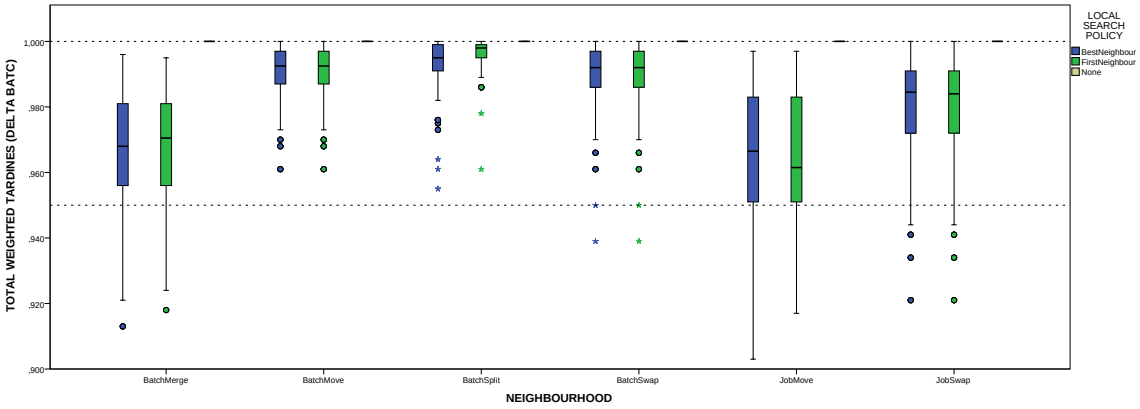


Figure 77: Total weighted tardiness improvement depending on the neighborhood structure combined with different local search scheme (experiment E.21)

metaheuristics search, the idea behind this can be described as the assumption that a single shaking move may not be sufficient to escape from the valley that contains the current local optimum.

However the results clearly show that multiple shaking moves do not improve the search procedure. Unfortunately it becomes obvious that multiple shaking moves seem to slightly worsen the search performance by leading to a slightly increased number of worse solutions.

The VNS schemes achieve the best results if the shaking phase performs a single random movement as originally described in the literature. Refer to Figure 78; cf. Appendix E.24.

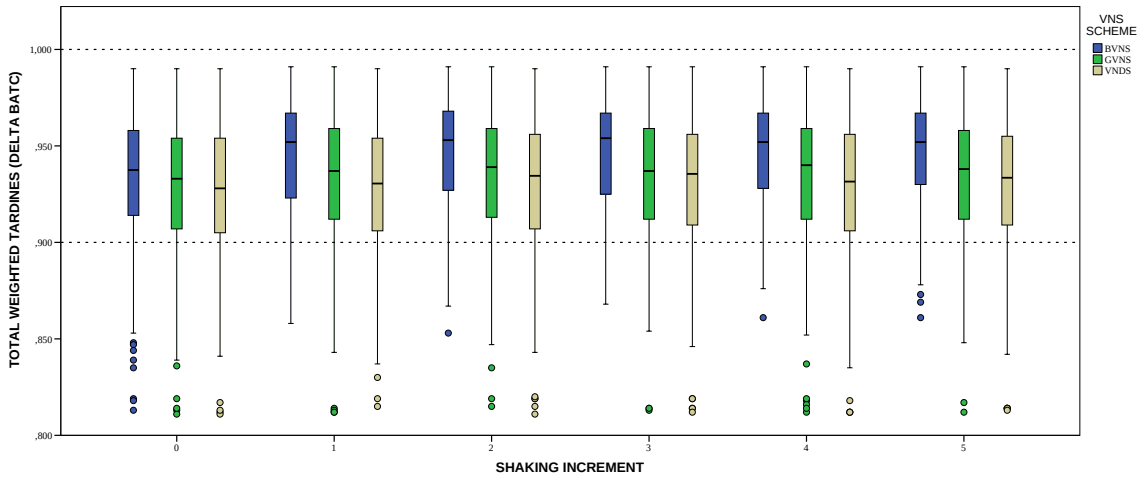


Figure 78: Total weighted tardiness improvement depending on the shaking range (experiment E.24)

8.3.5 Computational Deadline

The maximal time a search algorithm is allowed to search for improved solutions clearly influences the quality of the solution. It is widely accepted and observed in experiments that the improvements gained by optimization enhance with increasing run time. This experiment examines the effect of the computational deadline, evaluating the improvements that can be achieved especially with longer run times.

The experiment E.25 covers method settings that allow the VNS schemes BVNS, GVNS, and VNDS to search for a given amount of time ranging from 2 to 20 minutes. The experiment E.26 describes a similar experiment, but with fewer variants of computing time levels ranging from one

minute to five minutes. However the experiment E.26 deals with much more replication runs for each design point; 50 replication runs are performed for each run in order to find out if the range of the quality of the solutions changes.

The observed improvements increase with increasing run time as expected, but not as strong as someone would expect. The difference between the shortest and the longest run time scenario is below 3%. Another interesting observation points to a difference in the search behavior between the search schemes. Generally BVNS is outperformed by GVNS and VNDS in all cases. Interestingly GVNS outperforms VNDS especially for shorter run times, whereas the difference between both seems to decrease with longer run times. Refer to Figure 79; cf. Appendix E.25.

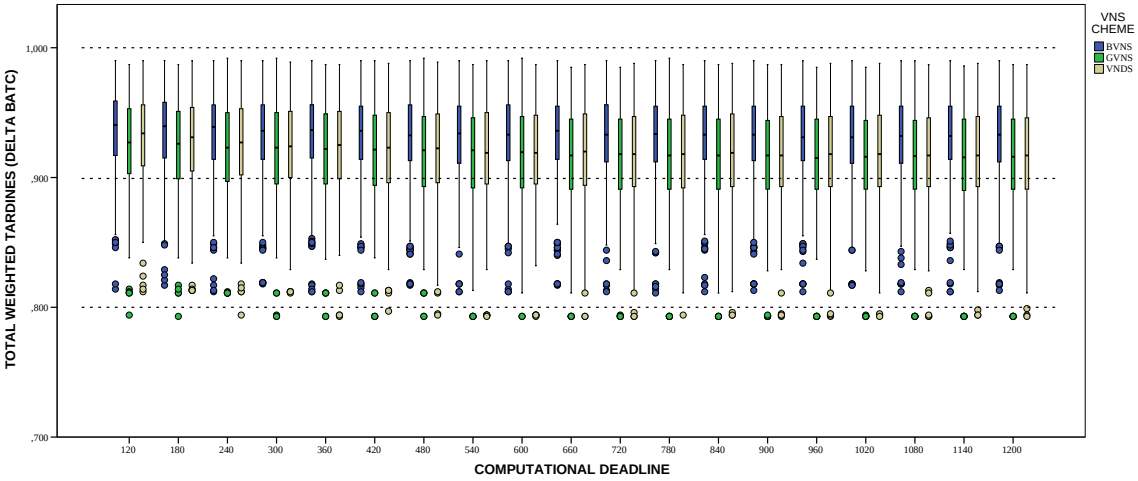


Figure 79: Total weighted tardiness improvement depending on the computational deadline (experiment E.25)

The results from the experiment E.26 show no obvious differences compared to the experiment E.25, despite of the fact that the experiment E.26 performs 10 times the replication runs of experiment E.25. The distributions of the solution quality levels remain unchanged, indicating that a significantly increased number of replication runs does not provide more insights.

Identical to the experiment E.25, the improvement by optimization is heightened with increasing computing time (in E.26). Interestingly the increase in improvement develops more strongly with computing time for the case with five machines compared to the cases with fewer machines. Refer to Figure 80; cf. Appendix E.26.

8.3.6 Search Space

In contrast to the other sections describing experiments, this section focuses on the search procedure carrying to analyzing the final solutions generated by the search schemes with specific settings. Knowledge about the problem structure as well as the search method's functioning facilitates the understanding of the search scheme and hereby helps to improve the algorithms further.

The experiment E.27 provides different views on the search behavior by showing traces along the search from the beginning to the end. It is an approach to visualize the search space and to describe the components that lead to enhanced solutions more in detail. The implementation is capable of tracing every single step during the search, e.g. identifying an optimum or performing a shaking step. 100 complete traces were created for the GVNS method with two different initial solutions, applied on the 90 benchmark models. The GVNS procedure is allowed to run 10 minutes, tracing all activities that lead to the final solution.

The diagrams in Appendix E.27 show a distribution of the visited solutions for the first five model instances in a coordinate system that is meant to represent the search space. It visualizes 1. the initial solution, 2. every local optimum, 3. the results of shaking steps, and 4. the final solutions for 100 runs for each model. The objective value that is drawn on the vertical axis and the horizontal axis describes a factor that represents the similarity of the solution to the best

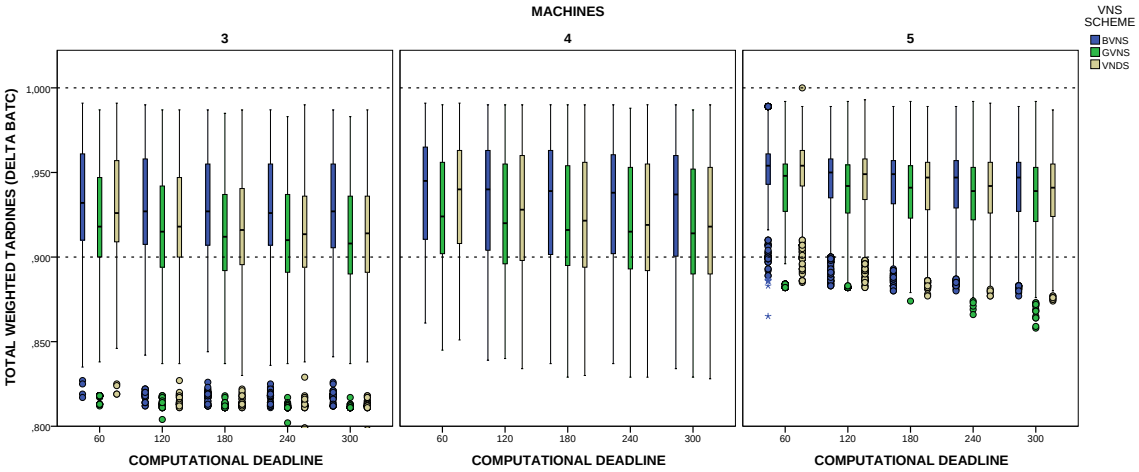


Figure 80: Total weighted tardiness improvement depending on the computational deadline (experiment E.26)

benchmark solution, i.e. the solution distance factor. The solution distance factor for two identical solutions equals one, which means that every job is scheduled on the same machine and all the jobs are sequenced in the same order. This way it is possible to visualize the distance from a particular solution to the best known solution taken from the benchmark data.

The diagrams show that the runs with the initial random solution (RND) show a wider distribution of solutions. Previous experiments show that it is recommended to use a more problem-specific policy such as the BATC rule. However using random as the initial solution does not lead to impaired solutions in every case. It depends on the model and the structure of the problem. The results show that optimization runs with a random solution can lead to better solutions than those solutions starting from a BATC solution in some cases. Compare the part in the diagrams in Appendix E.27 that relates to the model identifier with number 3.

Another interesting observation is that the search space contains solutions of good quality (comparable to the optimal solutions) that show relatively high solution distance factors, which means that optimal solution (best known) is not very similar. It is interesting to know that nearly optimal solutions do not necessarily have to be strongly similar to the optimal solution.

Figure 81 shows the number of improvement moves that are required to identify an optimum. It is easy to see that identifying the first optimum (starting from the initial solution) requires considerably more moves than finding the following optima later during at search. It takes minimum 20 improvement moves to settle at the first optimum. From then on it takes a maximum of 20 moves to get from one optimum to the next.

The number of moves required to reach the next optimum decreases with the increasing optimum level in the diagram. But this does not necessarily mean that the more local optima are found fewer moves are required to move to the next. This is presumably not the case, assuming that the effect in the diagram does not correctly represents the search behavior during the end of the search; the number of analyzed moves most probably decreases due to the search beeing terminated after 10 minutes and therefore a small number of runs actually achieves more than 10 optima in sequence. Consequently the number of improvement moves depicted in the diagram decreases with the local optima found. However the diagram in Figure 81 provides strong indication that finding an optimum in the early phase of search requires a quite stable number of improvement moves.

Furthermore the diagram in Figure 81 shows no difference for the two different initial solutions (BATC and RND) with one exception: identifying the first optimum takes fewer moves when the initial solution is problem-specific such as BATC rather than random (RND).

Figure 82 shows the number of shaking steps that are required to get from one optimum to the next. Clearly it requires no shaking step to identify the first optimum, which is the direct result of the first LS procedure.

In the early phase of the search it can be stated that identifying the third optimum requires

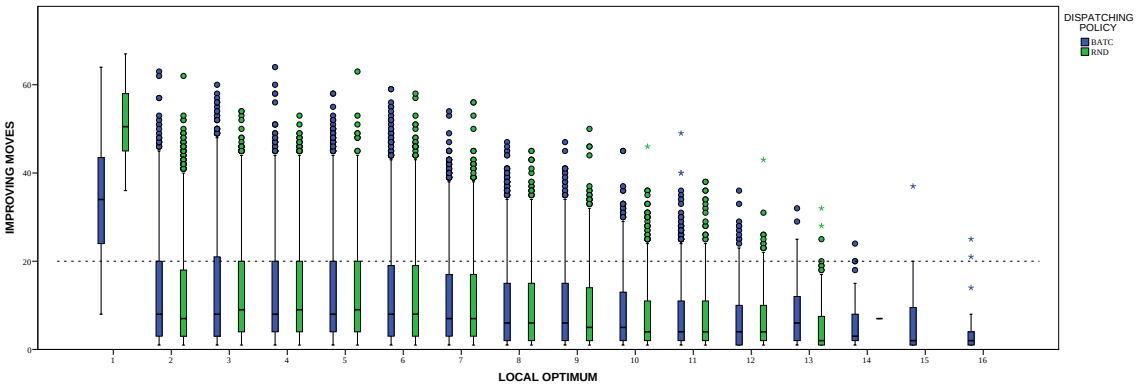


Figure 81: The number of improving moves required to get to the next optimum (experiment E.27)

more shakes than it takes finding the second optimum. From then on the number of required shakes continuously decreases. The reason for the decrease is the same as for the number of improvement moves that decrease with the number of optima found: the search is terminated after 10 minutes and therefore fewer runs reach more than 10 optima. Refer to Figure 82.

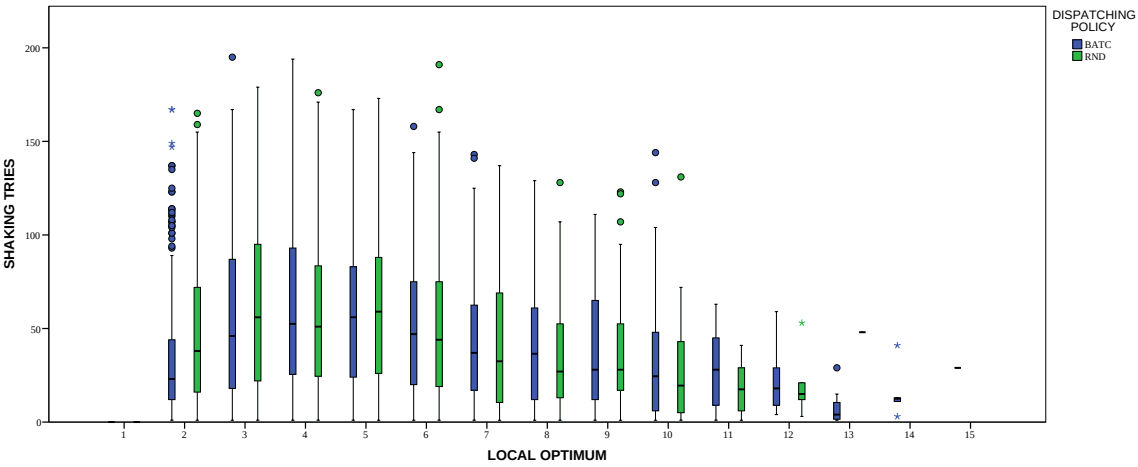
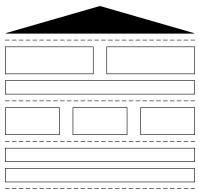


Figure 82: The number of shaking tries required to get to the next optimum (experiment E.27)



9 Conclusions and Outlooks

Contents

The core of this work revolves around the implementation of a scheduling framework developed to deploy it as an operational batch scheduling system in the diffusion and oxidation area of a waferfab frontend. This thesis covers both aspects from theory and practice in nearly equal measures. It provides the underlying theoretical background and reports valuable practical experiences from implementing a scheduling system into a real-world industrial environment.

The focus of the theoretical part lies on two topics. *a*) First this work provides an extensive review about the current state-of-the-art in the area of batch scheduling research. *b*) Second the theoretical groundwork of this work comprises a detailed analysis of the complexity status of the most common batch scheduling problems.

The focus of the practical work is located between the poles of academia and industry. The implemented framework is a balancing act between academia and industry since it basically comprises two systems: *a*) the experimental system and *b*) the prototype system. On one hand the framework satisfies the needs from academia with an experimental system that offers the capability to properly investigate academic questions in the area of metaheuristic batch scheduling. On the other hand the framework contains a prototype that is purposefully designed and developed to satisfy the needs posed by industry. The original intention of the framework's design is to provide a functioning prototype that is suitable to run as a real-time scheduling system on the operational level.

The main results of this work and related outlooks are organized in three main topic areas: *a*) theoretical background and state-of-the-art (Section 9.1), *b*) valuable insights spawned by experiments (Section 9.2), and *c*) experiences from implementing a prototype (Section 9.3).

9.1 Theoretical Background and State-of-the-Art

The groundwork for this thesis and the basis for implementing the scheduling framework is mainly build of *a*) a literature review about batch scheduling, *b*) the state-of-the-art for metaheuristics, and *c*) a review on the complexity status of the most common batch scheduling problems.

An extensive literature review about batch scheduling in wafer fabrication stands as one of the main pillars of this work (Section 2). The theoretical groundwork is complemented with a short overview on the state-of-the-art in metaheuristic optimization (Section 5). A detailed introduction to the batch scheduling topic with an detailed analysis of the complexity results of the most common batch scheduling problems completes the theoretical background of this work (Section 6).

Literature Review about Batch Scheduling The vast literature review about batch scheduling problems and solution approaches in Section 2 provides a representative overview on the state-of-the-art in this field. The review covers 170 publications in total, methodically structured in 16 groups representing different combinations of constraints for the single and the parallel machine environment. Refer to the overview tables in Appendix A to see all researched literature sources at a glance. Section 6 does investigate the current state of research in batch scheduling further, more thoroughly examining the different solution approaches to batch scheduling problems.

The literature provides a variety of simple and more sophisticated rule-based approaches that are easy to implement and that lead to acceptable solutions of good quality. These priority-based heuristics offer themselves for being directly implemented in dispatching systems with relatively low effort.

Beyond simple rules there are various studies where exact methods such as MIP, DP, CP, and B&B solve batch scheduling problems, whereby DP and B&B steadily become less important

and MIP formulations seem to be the most promising approach due to the success of powerful commercial software packages. The use of CP in this field is in the early stages and therefore poses as one of the interesting areas of future research. However all exact methods have in common that their use is limited to rather small instances and so are not advisable to be adopted in real-world implementations. Appendix C contains an extensive overview of run times of exact methods for common batch scheduling problems.

The literature complementarily provides a considerable amount of solution variants based on metaheuristics. The majority of solutions employ ACO, SA or GAs, whereas solution schemes based on VNS represent a minority. Unfortunately the literature provides no clear indication whether a specific metaheuristic generally performs better than another. By analyzing only the experimental results reported in the literature it is not possible to identify a superior metaheuristic for batch scheduling problems. As a matter of fact metaheuristic search schemes outperform simple rules by design, i.e. by investing computational effort. Interestingly there are reliable points of reference saying that metaheuristics outperform exact methods, which are most often represented by the state-of-the-art solver CPLEX.

A synopsis of the literature review reveals some important perspectives for future work. It is striking that a minority of works deals with multiple objectives that are a characteristic for typical scheduling problems present in waferfabs, especially for those with BPMs. Another important aspect refers to critical constraints such as time bounds, which can lead to invalid solutions; this aspect remains underrepresented in literature and is clearly demanding for proper strategies in the real world. Despite of the fact that many authors compare different methods in their works, there is a glaring lack of public benchmark instances with high practical relevance that offer the opportunity to compare different methods under real-world conditions. This would enable practitioners to identify the advantages and disadvantages of different methods in certain use cases more effectively. Finally it is undeniably the case that technical documentations including contemplations of economical benefits about successfully installed implementations for scheduling problems in the real world are still considered as an exception, albeit with growing tendencies. A rising number of such reports would create confidence, accelerating the success of scheduling solutions in the industries.

State-of-the-Art for Metaheuristics It is popularly accepted in the OR community that the performance of metaheuristics is closely linked to the structure of the underlying search space. The interaction of a metaheuristic with the problem's underlying fitness landscape defines its performance, whereby the knowledge of the fitness landscape's structure facilitates developing effective metaheuristics. Unfortunately there is a lack of knowledge and empirical evidence to determine landscape features in the area of OR. The experiment in Section 8.3.6 describes an attempt to determine the search space of problem instances. Further efforts in this field, developing meaningful concepts of the search space and gathering more information about its structure, would help to develop better metaheuristics.

The idea that effective metaheuristics balance intensification and diversification efforts in their search behavior is a widely shared notion today. These two forces naturally act contrary to each other, but also complement each other at the same time. Intensification (exploitation) relates to metaheuristic components/activities that aim at (intensively) searching for new optima in a certain area of the search space. Diversification (exploration) means that a metaheuristic is capable of searching a maximum number of different regions of the search space. Developing a better understanding of the interaction between the two forces hints to a promising direction of future research.

Recent developments in metaheuristic research consider hybridization and parallelization as promising approaches, leading to an improved search performance. Hybrid metaheuristics refer to the idea of combining metaheuristics with other techniques for optimization. Hybridization aims to exploit the complementary character of different optimization strategies whereas parallelism can help to reduce the computation time and increase the solution quality by using the technology of distributed computing systems. Especially the concept of parallelization seems to offer great opportunities for method enhancements in the future, particularly in the light of rapidly growing computing power.

The NFLTs roughly say that the average performance of any pair of algorithms across all possible

problems is identical, even if one of them is random search. This means that no metaheuristic generally performs better than another at the outset, but a certain implementation of a metaheuristic can outperform another for a distinct set of problems; cf. Section 5.7.2.

The dominant procedure in developing new search methods in OR is comparative/competitive testing. The performance of an algorithm depends on various aspects, e.g. the fine tuning of parameters and the actual implementation. It is argued that the strategy of competitive testing lacks deeper insights into cause-effect relationships between an algorithm and its performance. In this context it can be said that explaining the performance is better than demonstrating performance; cf. Section 5.7.3. It is justifiable for the stated reasons to ask for more explaining studies in the future, which focus on the internal functioning of search methods. Refer to Section 5 for more detailed information about the state-of-the-art of metaheuristics and Section 8 for a description of the experiments carried out.

Complexity Status of Batch Scheduling Problems A significant amount of research in the area of complexity theory has focused on boundaries of polynomial time and $\mathcal{NP}$-hard problems. Section 6 and the table in Appendix B provides an overview on the entire set of $p-batch$ problem variants and their complexity status.

The complexity status is determined for almost all problems with a few exceptions. Most of the focused batch scheduling problems belong to the class of $\mathcal{NP}$-hard problems, whereas a minority is polynomially solvable. The complexity status of the problems $1 \mid p-batch, b<n \mid \sum C_j$, $1 \mid p-batch, b<n \mid \sum w_j C_j$, and $P \mid p-batch, b<n \mid \sum C_j$ still remains open.

The processing time model is a scheduling model's decisive characteristic determining its complexity. The longest job processing time model ($ljpt$) can be seen as the standard model if no further constraining information is provided. The family processing time model (fpt) is usually accompanied by the constraint of incompatible job families ($fmls$). But the constraint $fmls$ does not always imply the family processing time model. Vice versa the missing incompatible job family constraint $fmls$ does not always imply the longest job processing time model ($ljpt$). The underlying model would be denoted clearer with an additional tag in the formal description, e.g. $ljpt$ and fpt in combination with $p-batch$. The complexity results show that the family processing time (fpt) model is easier to solve than the $ljpt$ model.

The problem $1 \mid p-batch, b<n, r_j \mid C_{max}$ is unary $\mathcal{NP}$-hard on the basis that the longest job processing time model ($ljpt$) is assumed. If the problem is subject to incompatible families, i.e. relies on the family processing time model (fpt), then the problem looses $\mathcal{NP}$-hardness; it is shown that the problem $1 \mid p-batch, b<n, r_j, fmls \mid C_{max}$ can be solved polynomially (if fpt is in use).

The problem $1 \mid p-batch, b<n \mid L_{max}$ is $\mathcal{NP}$-hard in the strong sense even if $b=2$; proven for the $ljpt$ model. But there is a polynomial algorithm for $1 \mid p-batch, b<n, fmls \mid L_{max}$, assuming identical processing times of jobs of the same family (fpt). This is quite in contrast to the complexity result for the original problem without job families ($1 \mid p-batch, b<n \mid L_{max}$) that is proven to be strongly $\mathcal{NP}$-hard when applying the the longest processing time model ($ljpt$).

9.2 Valuable Insights Generated by Experiments

Experimentation is an indispensable tool for understanding and subsequently improving a system and methods. The implemented framework is designed to be used as an experimental system that offers the capability to properly investigate academic questions. It offers the capabilities to analyze the effect of the framework's system factors on the system performance in a reproducible environment.

The intended use as an experimental system poses many different requirements to the design and the development of the system. Carrying out a larger number of experiments requires sophisticated capabilities in model generation, execution management, and the analysis of experimental results. Especially the execution management of thousands of optimization runs on high-performance machines with multiple cores requires elaborate data management capabilities. Loading various benchmark model instances requires additional effort during the development besides the substantial efforts in establishing the ability for generating and utilizing user-defined models.

One of the key tasks of the experimental system is to determine the benefit which could be expected from deploying a scheduling system in the real world. The experiments serve to provide

reliable numbers for economic benefits based on which a manager decides whether installing a scheduling system is profitable. Unfortunately the results of simulation and optimization studies depend on the initial assumptions to a large extend. The benefit in numbers is often not a question of the internal system's performance rather than a question of the initial experimental settings. The experiments show that even slight changes in the experimental setup can result in considerable changes of the output. Researchers tend to highlight the accomplishments and the merits of their systems/methods while often understating the disadvantageous aspects observed during experimenting. The results of any experiment need to be treated with caution since the absolute numbers always mirror a certain set of experimental settings only. Provisioning a wide range of public benchmark instances with high practical relevance would help the evaluation of different scheduling approaches in distinct situations in the future.

One of the main goals in the scheduling community is to define new effective search strategies, identifying effective algorithmic components as parts of superior search schemes. It is common practice to ascribe the observed improvements in the objective value to the search method and its algorithmic improvements. But the obvious question is that of the sources of the observed improvements considered as optimization effects. The experiments pose the question whether the problem instance's characteristics or the scheduling method settings have a greater impact on the improvements. A promising aspect of future works would be to investigate the effect of important control parameters further along in order to clarify the sources of vast optimization potentials.

For designing the experiments it is of importance to know the time a LS scheme requires to identify a local optimum. The complexity of a problem mirrored by the run time of a deterministic LS scheme such as VND depends on the objective function as well as on the number of machines and jobs. The experience with the run time results and the knowledge about local optima helps to determine the size of model instances. Proper model sizes in combination with suitable computational deadlines ensure that the observations reflect the search behavior in its entirety.

The most important results and valuable insights spawned by the experiments are summarized in the following two paragraphs: *a*) insights related to model characteristics (Section 9.2.2) and *b*) insights related to method settings (Section 9.2.1).

9.2.1 Insights Related to Model Characteristics

The experiments pose the question whether the problem instance's characteristics or the scheduling method settings have a greater influence on the improvements. Confronted with this question several experiments were carried out in order to identify the origins of the observations better. The described experiments investigate the effect of the structure of a model instance with its model characteristics on the objective function. They are designed to identify the main contributors to the enhancements gained with the optimization procedure. It turns out that the structure of the model instance with its characteristics has a major influence on the optimization results. Some model characteristics promote high optimization values while others only lead to minor enhancements.

An interesting question for future studies would be to find out if the observations in this work can be confirmed by further studies in identical or similar scheduling environments. It would be of further interest whether similar observations can be made for another optimization method beyond VNS, e.g. for other metaheuristics or even for the class of exact methods.

Objective Function The experiments reveal considerable differences regarding the benefits created for distinct objective functions. For example minimizing TT generally shows greater improvements in numbers than minimizing TCT. The TT improvements rise up to 40% whereas the biggest improvements in TCT do not exceed 20% compared to the referenced solutions that are both created with very simple dispatching rules, i.e. EDD (TT) and SPT (TCT).

Another result refers to the experimental run time data from which conclusions regarding the complexity of objective functions can be drawn. Given that the run time of the deterministic LS routine VND serves as a point of reference for the problem's complexity it is possible to compare the complexities of problems. The run times indicate for example that the problems with the objective TCT are harder to solve than those with the objective C_{max}, although no complexity reduction rule exists between both. The run time results also make it obvious that

minimizing TT and TU takes more time than minimizing L_{max}, underpinning the fact that minimizing L_{max} is the easiest problem between these three with respect to the findings from the complexity theory. But the run time results provide no indication that solving problems with TU is easier than those with TT or vice versa. There are no reduction rules in place for their relationship but the experiments show that the initial due date setting influences their run times and hereby gives indication when it is easier to minimize TT or TU.

Refer to Section 8.1.1 and Section 8.1.9 for a discussion on the corresponding experiments.

Number of Machines The experiments indicate that the number of machines has a relatively strong effect on the optimization results. The results clearly indicate that a higher number of machines leads to greater improvements in the objective as long as the size of the problem remains manageable. Therefore the conclusion can be drawn that scheduling a work center is more worthwile when it contains more machines. It is likely that this observation can be reduced to the the partitioning effect of the scheduling procedure. In contrast the classical dispatching approach does not look across machines and subsequently lacks finding a proper mapping between machines in parallel and the jobs waiting in front of them. Refer to Section 8.1.2 for a discussion on the corresponding experiments.

Number of Jobs As it is with the number of machines the problem apparently becomes harder to solve with an increasing number of alternatives resulting from more jobs. But the experiments indicate that the number of jobs does not have tremendous effect on the optimization results compared to the number of machines. However the number of jobs has a stronger effect on the run times than the number of the machines. This observation probably serves as an indication that sequencing is computational more expensive than partitioning. Refer to Section 8.1.3 for a discussion on the corresponding experiments.

Number of Job Families The number of job families, as one of the main factors, has a considerable effect on the search behavior. Interestingly the search behaviors observed in conjunction with changing numbers of job families depend on the objective. The experimental results show that the number of job families has no significant effects on the objective TCT. In contrast the search behavior observed for the objective TT considerably depends on the number of job families. It can be observed that the improvement in TT increases with the number of job families in the case of four machines. Interestingly this behavior turns around in the case of additional machines, e.g. for the case of 12 and 16 machines the TT improvement decreases with the number of job families.

Refer to Section 8.1.4 for a discussion on the corresponding experiments. The effect of the job families is discussed further in the context of batch sizes (Section 8.1.6), machine eligibility constraints (Section 8.1.7), and non-identical job sizes (Section 8.1.8).

Process Time The processing time is naturally one of the most decisive factors. The experimental results suggest that the process time scheme influences the level of improvement in a way that a higher variance in processing time causes greater improvements in TCT. The same effect, albeit weaker, is observable for the objective TT. The greatest improvements are gained for the cases with highly variant processing times, which seem to assist greater improvements by increasing the number alternatives in the scheduling decisions. The experiments clearly show that this effect is stronger for the objective TCT than for the objective TT. Refer to Section 8.1.5 for a discussion on the corresponding experiments.

Batch Sizes The experiments show that the improvements both in TCT and TT strictly increase with increasing batch sizes; vice versa, the improvements decrease with smaller batch sizes. The improvement for the scenarios with the smallest batch sizes tend to zero. Despite of the fact the optimization procedures for TCT and TT behave similar with respect to the improvement in the objectives, there are differences in the run time behavior. The run time for the objective TT increases with increasing batch size whereas the run time for the objective TCT shows the greatest values for the smallest batch sizes. Refer to Section 8.1.6 for a discussion on the corresponding experiments.

Dedication Density The experiments expose that the effect of the dedication density also depends on the objective, saying that the observed behavior for the objective TCT differs from that documented for the objective TT. The average of the TCT improvement increases with increasing density factor, whereas the average TT improvement decreases with increasing density factor. This means that a uniform work center without any process dedications provides the best environment to minimize CTs by optimization. In contrast a higher number of process restrictions supports TT optimization. Refer to Section 8.1.7 for a discussion on the corresponding experiment.

Job Sizes Similarly to the effect observed for the dedication scheme, the search method behaves contrary for the objectives TT and TCT with respect to different job size models. The experiments show that if more small lots need to be scheduled more improvement in CT can be expected by optimization. The behavior turns around when the goal is TT minimization in a sense that the improvement in TT increases with increasing job size in most cases. Put it another way this observation indicates that minimizing tardiness leads to higher improvements in the case of less small jobs. Refer to Section 8.1.8 for a discussion on the corresponding experiment.

Initial Due Dates The experiment shows that the initial setting of the due dates considerable affects the optimization results. Defining the distribution from which the initial due dates are drawn during the model generation procedure implicitly determines the improvements gained by optimization. In fact there are unfavorable model settings existing for which no optimization method will probably generate acceptable results suitable for discussion. The experiment shows that a due date mean value between zero and twelve hours shows the best results for the objectives TT and TU. The improvement generally decreases in the area of the extremes, i.e. when the due dates are too tight or too loose. Generally the model scenarios with the lowest variance in the initial tardiness show the highest improvements in all cases. Refer to Section 8.1.9 for a discussion on the corresponding experiment.

Job Priorities Job priorities play a very important role for the scheduling of waferfabs. The experiment shows that the improvement in TWCT and TWT increases with increasing number of priority groups, albeit slightly. On the contrary the priority weight factor shows no clear pattern of influence. Refer to Section 8.1.10 for a discussion on the corresponding experiment.

Correlation between Objectives One of the main differences between the scheduling problems in the real world and academic scheduling studies is that the real-world needs to manage multi-objective functions. The experiments provide a correlation study between the basic objectives. The experimental results show four main observations: *a*) The basic objectives correlate with their weighted versions (0.8+). *b*) The OTD measures correlate, but L_{max} is exceptional (0.7+). *c*) The TCT and the C_{max} share a loose relationship (0.5-0.6). *d*) The TCT slightly improves OTD measures (0.3-0.4). Refer to Section 8.1.11 for a discussion on the corresponding experiment.

Pareto Optimization Section 8.1.12 discusses an experiment that is designed to analyze bi-criteria objective functions, which combine two criteria equivalent to each other (pareto scheme). The experiments show that the two objectives either agree or disagree, whereas some combinations behave neutrally. The combined pareto objective often orientates on one of both objectives and some pareto objectives nearly behave identical to a single objective. Multi-objective optimization is still an underrepresented type of problem that deserves more attention.

Look-Ahead Horizon The results show that information about arriving jobs empowers the scheduling method remarkably and that the improvement tends to steadily increase with an increasing look-ahead horizon below a certain threshold of several hours. For look-ahead horizon values greater than this threshold the improvement degrades due to the fact that the problem instances become too large. Refer to Section 8.2.2 for a discussion on the corresponding experiments.

Arrival Errors An important conclusion drawn from the experiments is that higher mean error values result in slightly lower improvements as expected. Interestingly the decrease in the improvement caused by arrival errors is very low. It is commonly accepted that look-ahead control strategies are not very sensitive to errors in job arrival predictions, implying that job arrival predictions do not need to be very accurate to be useful. Refer to Section 8.2.3 for a discussion on the corresponding experiments.

9.2.2 Insights Related to Method Settings

The OR community thrives on the theory that no metaheuristic generally performs better than another at the outset, though a certain implementation of a metaheuristic can outperform another for a distinct set of problems (NFLTs); cf. Section 5.7.2. This means that a search scheme designed for a specific type of problem needs to be adjusted with favorable settings. A favorable setting is one that balances intensification and diversification efforts in the search behavior. This section summarizes the results from the experiments carried out to identify a superior VNS scheme by investigating the effect of basic search components and settings. Refer to Section 5 for more detailed information about the state-of-the-art of metaheuristics and Section 7.2 for a description of the implemented VNS variants in this framework.

As a result from summarizing the results it is possible to conclude on a few aspects pointing to interesting directions of future research. For example, it would be interesting to know if the observations made for the batch scheduling problem in this work can be made for other scheduling problems with comparable results. Another unresolved issue refers to the actual implementation of the neighborhoods and the search schemes on the code level; very little is known about the effect of certain programming concepts and data structures, which in fact determine the performance of each implementation of a certain search scheme.

Search Schemes The experiments reveal considerable differences in performance for the basic VNS variants studied in this work. The VNS concept comprises five basic search schemes, e.g. GVNS, VNDS, BVNS, VND and RVNS. GVNS and VNDS lead to the best results followed by BVNS and VND in that order; the variant RVNS shows the worst results. Generally BVNS is outperformed by GVNS and VNDS in almost every case. Interestingly GVNS outperforms VNDS especially during shorter run times, whereas the difference between both seems to decrease with longer run times. However in some cases VNDS performs equally or even better than GVNS. Refer to Section 8.3.2, Section 8.3.4, and Section 8.3.5 for discussions on the corresponding experiments.

Initial Solution An important outcome of the experiments is that the initial solution has a considerable effect on the VNS method's performance, though it is not that important as often discussed in the literature. A considerable amount of experiments deals with the BATC dispatching rule. The BATC rule requires determination of a control parameter k and the experiments show that VNS performs best when the k value is between 0.4 and 0.6. The results indicate that the VNS search scheme is more important than the initial BATC solution. Refer to Section 8.3.1 for a discussion on the corresponding experiment.

Neighborhoods The structure of the neighborhoods is one of the most urgent issues to solve when designing a VNS procedure. The results show that some neighborhood combinations perform better than others. The best results are achieved with those VNS variants that move jobs, swap jobs, or merge batches. On the contrary splitting, moving and swapping batches leads to less favorable results. One experiment evaluates the combinations of three pairs of two neighborhoods. Better results occur for neighborhood combinations involving the two pairs split/merge batches and move/swap jobs. Between those the best results occur if the second search level operates on the job level, i.e. uses job swapping and job moving.

The neighborhood sequence has nearly no effect on the performance of the VNS procedures. There is a very small advantage for those method settings that start their search on the second level with moving and swapping jobs. One experiment tests the VND scheme with 720 neighborhood sequences, which simply represent all possible ordered sequences of the six neighborhoods. The difference between all these combinations of neighborhood sequences

records a maximum improvement of 1%. Refer to Section 8.3.2 for a discussion on the corresponding experiments.

Local Search (LS) Finding proper settings for the LS components within metaheuristics is another critical issue that determines the search method's performance. The experiments show no clear difference in performance between the two classical LS schemes under study, i.e. between best-improvement search and first-improvement search. The only difference is the number of moves the methods perform until the computational deadline is reached. The first-improvement LS scheme performs more moves than the best-improvement LS scheme, which indicates that the first-improvement LS scheme is capable of exploring more search space. However the first-improvement strategy does not outperform the best-improvement strategy in terms of solution quality.

When comparing the run times it becomes obvious that the best-improvement procedure needs more time than the first-improvement method. However their performance in terms of solution quality is nearly identical. Refer to Section 8.3.3 for a discussion on the corresponding experiments.

Shaking Shaking is another crucial aspect regarding the design of VNS methods. The basic VNS schemes specify a single shaking step in the perturbation phase. The experimental results clearly show that multiple shaking moves do not improve the search procedure. Unfortunately it becomes obvious that multiple shaking moves seem to impair the search performance slightly by leading to a marginally increased number of worse solutions. It turns out that the VNS schemes achieve the best results if the shaking phase performs a single random movement as originally described in the literature. Refer to Section 8.3.4 for a discussion on the corresponding experiment.

Computational Deadline Assessing an appropriate computational deadline is a key factor to the settings of any metaheuristic, presupposing data-founded knowledge about the size of the underlying scheduling problem. The experiments show that the observed improvements rise with increasing run time as expected, but not as strong as someone would expect. The difference between the shortest and the longest run time scenario lies below 3%. Refer to Section 8.3.5 for a discussion on the corresponding experiment.

Time Window Decomposition (TWD) TWD is used to disassemble scheduling problems on the timeline into smaller sub problems. The time window size is a decisive factor that influences the optimization results. The experiments show that the CT tends to increase with an increasing time window size.

But the results also show that the smallest time window does not necessarily lead to the best objective value. There are values existing for the time window size that lead to considerably better results than the smallest possible time unit does. Even slight changes in the time window interval, e.g. adjacent time window intervals, can lead to totally different solutions.

Moreover it turned out that applying an optimization technique in conjunction with TWD does not always lead to improvements. The experiments show the phenomenon that some values for the time window interval lead to solutions outperformed by their corresponding dispatching reference. There are some unfavorable experimental settings that can lead to negative changes in the objective value compared to the referenced dispatching solution in a few cases. In this special case the pure dispatching solution performs better than optimization combined with TWD. Refer to Section 8.2.1 and Section 8.2.2 for a discussion on the corresponding experiments.

9.3 Valuable Experiences from Implementing the Prototype

The second main intention of the framework's design besides the capability of running experiments is to provide a functioning prototype that is suitable to serve as a real-time scheduling system on the operational level. The prototype is purposefully designed and developed for the needs posed by the industry.

Unfortunately the problems in academia and industry are rarely identical. The prototype code and the code for the experimental system share basic core functionalities, but fairly large parts of both are made of very problem-specific implementations. The practical problems from industry are typically afflicted with many different constraints. It is often necessary to design case-specific solutions for some special constraints beyond those standard constraints reported in the literature. The objective function usually comprises multiple objectives, whereby some of them can have a special character and do not belong to the standard measures typically used. Academia and industry would benefit from more detailed descriptions of urgent industrial problems; these descriptions would assist finding better solutions for the real world in the future.

Designing, implementing and testing a scheduling system is a tough task, but deploying it in a waferfab that relies on dispatching to that date is even harder. The scheduling framework essentially comprises two main components: *a*) the top-level scheduling system with all its modeling, simulation and optimization functionalities and *b*) the underlying data level connected to the waferfab's MES. Both parts, the top-level system with its sophisticated scheduling capabilities and the underlying data level are causing approximately equal amounts of work during the development of the entire system.

The following enumeration summarizes the main experiences and lessons drawn from the development of the prototype.

Data Structure and Data Flow The massive amounts of data can be roughly divided into *a*) static data (master data) and *b*) dynamic data (snapshot data). Static data refers to information that changes rarely such as basic machine information or process flows. Dynamic data refers to objects in manufacturing that change their state more frequently such as lots. The table structure comprises twelve tables structured into three groups hierarchically connected via primary and foreign keys, which enables a cascading deletion of related data sets in different tables.

Establishing the data transfer from the MES to the scheduling database is one of the most time-consuming activities during the development and the deployment of a scheduling system. The data that is required to create a model from the scheduling problem is often distributed in many different data tables of the MES and other related databases. The high complexity of the data infrastructure in combination with the huge volume of data leads to considerable demand of manpower developing proper software and managing data with all its problems. The data flow of the framework is based on dozens of data reports that extract, filter, analyze and merge data sets from different sources.

Loading and Validating Models The scheduling procedure begins with ascertaining the status of the machines and the jobs on the data level. Their current status combined with the master data basically form the scheduling problem. A series of data reports is arranged in a kind of master report managing the execution of the subordinated data reports in the intended order.

The extracted snapshot data in the database does not always exactly represent the status of the shop floor. The real-world data sets are subject to errors and validation procedures need to be installed in order to derive valid model instances from the data. Data errors in a typical scheduling project can be traced back to one of the three classified error sources: *a*) the complexity of the data infrastructure, *b*) the human element, and *c*) excursions on the shop floor.

Data problems are considered to be a major obstacle while transferring results from academia into a real-world application. Therefore researchers focus on techniques related to data validation and data mining in order to deal with missing or erroneous data sets. This framework uses an object-oriented data validation procedure, ensuring that the input data satisfies the data quality standards.

Exception Handling on the Shop Floor An operational scheduling system in industry must ensure that the result of the computation is always a valid schedule. Since the real-world data comes with errors proper validation procedures need to be implemented. This is an important point especially in the case of critical constraints and strict computation time limits. Nevertheless a scheduling system (as well as a dispatching system) needs decent exception handling strategies that manage dysfunctions on the shop floor.

Experts' Support Developing a scheduling system is a tough task, but deploying a scheduling system in the real world is another matter. The team needs reliable assistance from all the people involved, from the shop floor level to the management level. Unfortunately a scheduling system comes in touch with nearly all experts in the area of factory operations. A scheduling project without serious support provided by the affected people will be doomed to fail. It is of essential importance to convince experts and their managers to support the scheduling project on all factory levels, especially on the data level and on the factory floor.

Equipment Models Equipment models form one of the central pillars when installing a scheduling system on the shop floor. Especially batch scheduling systems benefit from look-ahead capabilities predicting job arrivals for a certain time horizon. But modeling wafer fabrication equipment in order to predict processing behavior is another difficult task. Conventional analytic equipment models used for long-term simulation applications usually consist of information about THP and may include additional information for BP. A scheduling system needs more detailed models that mimic the equipment's processing behavior more precisely. Especially predicting accurate processing times comes into focal points in the field of equipment modeling.

Optimization Method One of the main components of a scheduling system is the optimization procedure that seeks for optimized schedules as a core component in the prototype. Once the decision for a optimization system is made the question is whether an exact or metaheuristic method comes into operation.

Many studies show that exact methods are still restricted to small use cases even when state-of-the-art solvers are in use. It is shown by various examples that metaheuristics can outperform exact solution schemes. Consequently exact methods need to be applied in combination with decomposition techniques, e.g. machine-based decomposition. At this point it is necessary to remember that the improvement by optimization depends considerably on the number of machines building the work center; the optimization potential rises with the number of machines. Reducing the work center's size by using machine based decomposition in order to apply exact methods will have a detracting effect in the opposite direction.

To our best knowledge there is no proprietary software existing that offers implementations of metaheuristics to be used for industrial applications. The only alternative there is then to develop a metaheuristic scheduling system in-house, which is probably not feasible due to internal guidelines or not possible due to the lack of qualified personnel. In contrast exact methods come with commercial solver packages supported by prestigious companies.

References

Samuli Aalto. Optimal control of batch service queues with finite service capacity and linear holding costs. *Mathematical Methods of Operations Research*, 51(2):263--285, 2000. ISSN 1432-2994. doi: 10.1007/s001860050088. URL http://dx.doi.org/10.1007/s001860050088.

Evon M. O. Abu-Taieh and Asim Abdel Rahman El Sheikh. Methodologies and approaches in discrete event simulation. In Asim Abdel Rahman El Sheikh, Abid Al Ajeeli, and Evon M. O. Abu-Taieh, editors, *Simulation and modeling*. IGI Pub., Hershey, 2008. ISBN 9781599042008.

David Adamy. *Introduction to electronic warfare modeling and simulation*. Artech House, Boston and MA, 2003. ISBN 1-58053-495-3.

M. A. Adibi, M. Zandieh, and M. Amiri. Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Systems with Applications*, 37(1):282--287, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2009.05.001. URL http://dx.doi.org/10.1016/j.eswa.2009.05.001.

Gaurav K. Agrawal and Sunderesh S. Heragu. A survey of automated material handling systems in 300-mm semiconductorfabs. *IEEE Transactions on Semiconductor Manufacturing*, 19(1): 112--120, 2006. ISSN 0894-6507. doi: 10.1109/TSM.2005.863217. URL http://dx.doi.org/10.1109/TSM.2005.863217.

Javad H. Ahmadi, Reza H. Ahmadi, Sriram Dasu, and Christopher S. Tang. Batching and scheduling jobs on batch and discrete processors. *Operations Research*, 40(4):750, 1992. URL http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=4482900&site=ehost-live.

Elif Akcali, Reha Uzsoy, David G. Hiscock, Anne L Moser, and Timothy J. Teyner. Alternative loading and dispatching policies for furnace operations in semiconductor manufacturing: a comparison by simulation. In J. A. Joines, Russell R. Barton, K. Kang, and Paul A. Fishwick, editors, *Proceedings of the Winter Simulation Conference*, volume 2 of *WSC '00*, pages 1428--1435, 2000. doi: 10.1109/WSC.2000.899121. URL http://dx.doi.org/10.1109/WSC.2000.899121.

Enrique Alba, editor. *Parallel metaheuristics: A new class of algorithms*. John Wiley, Hoboken and NJ, 2005. ISBN 0-471-67806-6.

Susanne Albers and Peter Brucker. The complexity of one-machine batching problems. *Discrete Applied Mathematics*, 47(2):87--107, 1993. ISSN 0166-218X. doi: 10.1016/0166-218X(93)90085-3. URL http://www.sciencedirect.com/science/article/pii/0166218X93900853.

Theodore T. Allen. *Introduction to discrete event simulation and agent-based modeling: Voting systems, health care, military, and manufacturing*. Springer, London and and New York, 2011. ISBN 978-0-85729-138-7.

Eric Allender, Michael C. Loui, and Kenneth W. Regan. Reducibility and completeness. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and theory of computation handbook*. CRC Press, Boca Raton [u.a.], 2010. ISBN 978-1-58488-822-2.

Christian Almeder and Lars Mönch. Metaheuristics for scheduling jobs with incompatible families on parallel batching machines. *Journal of the Operational Research Society*, 62(12):2083--2096, 2011.

Davide Anghinolfi and Massimo Paolucci. Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers & Operations Research*, 34(11):3471--3490, 2007. ISSN 0305-0548. doi: 10.1016/j.cor.2006.02.009. URL http://dx.doi.org/10.1016/j.cor.2006.02.009.

Daniel Ashlock. *Evolutionary computation for modeling and optimization*. Springer, New York, 2006. ISBN 978-0387-22196-0.

S. S. Aurand and P. J. Miller. The operating curve: a method to measure and benchmark manufacturing line productivity. In *Advanced Semiconductor Manufacturing Conference and Workshop, 1997. IEEE/SEMI*, pages 391--397, 1997. doi: 10.1109/ASMC.1997.630768. URL http://dx.doi.org/10.1109/ASMC.1997.630768.

A. N. Avramidis, K. J. Healy, and Reha Uzsoy. Control of a batch-processing machine: A computational approach. *International Journal of Production Research*, 36(11):3167--3181, 1998. doi: 10.1080/002075498192355. URL http://dx.doi.org/10.1080/002075498192355.

Mehmet E. Aydin and Mehmet Sevkli. Sequential and parallel variable neighborhood search algorithms for job shop scheduling. In Fatos Xhafa and Ajith Abraham, editors, *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*, volume 128 of *Studies in computational intelligence*, pages 125--144. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78984-0. URL http://dx.doi.org/10.1007/978-3-540-78985-7_6.

Zahra Azimi. Comparison of metaheuristic algorithms for examination timetabling problem. *Journal of Applied Mathematics and Computing*, 16(1):337--354, 2004. ISSN 1598-5865. doi: 10.1007/BF02936173. URL http://dx.doi.org/10.1007/BF02936173.

Meral Azizoglu and Scott Webster. Scheduling a batch processing machine with non-identical job sizes. *International Journal of Production Research*, 38(10):2173--2184, 2000. doi: 10.1080/00207540050028034. URL http://dx.doi.org/10.1080/00207540050028034.

Meral Azizoglu and Scott Webster. Scheduling a batch processing machine with incompatible job families. *Computers & Industrial Engineering*, 39(3--4):325--335, 2001. ISSN 0360-8352. doi: 10.1016/S0360-8352(01)00009-2. URL http://dx.doi.org/10.1016/S0360-8352(01)00009-2.

Thomas Bäck. Fundamental concepts of evolutionary computation. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997a. ISBN 0-7503-0392-1.

Thomas Bäck. Self-adaptation. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997b. ISBN 0-7503-0392-1.

P. Backus, M. Janakiram, S. Mowzoon, C. Runger, and A. Bhargava. Factory cycle-time prediction with a data-mining approach. *IEEE Transactions on Semiconductor Manufacturing*, 19(2): 252--258, 2006. ISSN 0894-6507. doi: 10.1109/TSM.2006.873400. URL http://dx.doi.org/10.1109/TSM.2006.873400.

Sugato Bagchi, Chen-Ritzo Ching-Hua, Sameer T. Shikalgar, and Michael Toner. A full-factory simulator as a daily decision-support tool for 300mm wafer fabrication productivity. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2021--2029. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

Malti Baghel, Shikha Agrawal, and Sanjay Silakari. Article: Survey of metaheuristic algorithms for combinatorial optimization. *International Journal of Computer Applications*, 58(19):21--31, 2012.

Hari Balasubramanian, Lars Mönch, John W. Fowler, and Michele E. Pfund. Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. *International Journal of Production Research*, 42(8):1621--1638, 2004. doi: 10.1080/00207540310001636994. URL http://dx.doi.org/10.1080/00207540310001636994.

Osman Balci. Validation, verification, and testing techniques throughout the life cycle of a simulation study. In J. D. Tew, M.S Manivannan, D. A. Sadowski, and Andrew F. Seila, editors, *Proceedings of the 26th Winter Simulation Conference*, pages 215--220, San Diego and CA and USA, 1994. Society for Computer Simulation International. ISBN 0-7803-2109-X. URL http://dl.acm.org/citation.cfm?id=193201.194018.

Osman Balci. Verification, validation, and accreditation. In D. J. Medeiros, E.F Watson, J.S Carson, and M.S Manivannan, editors, *Proceedings of the 30th Winter Simulation Conference*, WSC '98, pages 41--44, Los Alamitos and CA and USA, 1998. IEEE Computer Society Press. ISBN 0-7803-5134-7. URL http://dl.acm.org/citation.cfm?id=293172.293183.

Osman Balci. A life cycle for modeling and simulation. *SIMULATION*, 88(7):870--883, 2012.

Catherine M. Banks. What is modeling and simulation? In John A. Sokolowski and Catherine M. Banks, editors, *Principles of modeling and simulation*, pages 3--23. John Wiley, Hoboken and N.J, 2009. ISBN 978-0-470-28943-3.

Philippe Baptiste. Batching identical jobs. *Mathematical Methods of Operations Research*, 52(3): 355--367, 2000. ISSN 1432-2994. doi: 10.1007/s001860000088. URL http://dx.doi.org/10.1007/s001860000088.

Philippe Baptiste, Peter Brucker, Sigrid Knust, and Vadim G Timkovsky. Ten notes on equal-processing-time scheduling. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(2):111--127, 2004. ISSN 1619-4500. doi: 10.1007/s10288-003-0024-4. URL http://dx.doi.org/10.1007/s10288-003-0024-4.

Amotz Bar-Noy, Sudipto Guha, Yoav Katz, Joseph Naor, Baruch Schieber, and Hadas Shachnai. Throughput maximization of real-time scheduling with batching. *ACM Trans. Algorithms*, 5(2): 18:1--18:17, 2009. ISSN 1549-6325. doi: 10.1145/1497290.1497294. URL http://doi.acm.org/10.1145/1497290.1497294.

Julio Barrera and Carlos A. Coello Coello. A review of particle swarm optimization methods used for multimodal optimization. In Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri, editors, *Innovations in swarm intelligence*. Springer, Berlin, 2009. ISBN 978-3-642-04224-9.

Roberto Battiti and Mauro Brunato. Reactive search optimization: Learning while optimizing. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Matthias Anton Becker. Modeling and simulation of a complete semiconductor manufacturing facility using petri nets. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA '03)*, volume 2, pages 153--155, 2003. doi: 10.1109/ETFA.2003.1248687. URL http://dx.doi.org/10.1109/ETFA.2003.1248687.

Matthias Anton Becker. Entwicklung eines dynamischen clustertool-simulators sowie entwicklung und bewertung von steuerungsregeln für parallele losbearbeitung für clustertools: Master thesis, 2007.

J. Behnamian, M. Zandieh, and S. M. T. Fatemi Ghomi. Parallel-machine scheduling problems with sequence-dependent setup times using an aco, sa and vns hybrid algorithm. *Expert Systems with Applications*, 36(6):9637--9644, 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2008.10.007. URL http://dx.doi.org/10.1016/j.eswa.2008.10.007.

N. Bengtsson, G. Shao, Björn Johansson, Y. T. Lee, S. Leong, Anders Skoogh, and C. Mclean. Input data management methodology for discrete event simulation. In M. D. Rossetti, Raymond R. Hill, Björn Johansson, Ann Dunkin, and Ricki G. Ingalls, editors, *Proceedings of the Winter Simulation Conference*, WSC '09, pages 1335--1344, 2009. URL http://dx.doi.org/10.1109/WSC.2009.5429651.

Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: an international journal*, 8(2):239--287, 2009. doi: 10.1007/s11047-008-9098-4. URL http://dx.doi.org/10.1007/s11047-008-9098-4.

Andrew Bilyk and Lars Mönch. A variable neighborhood search approach for planning and scheduling of jobs on unrelated parallel machines. *Journal of Intelligent Manufacturing*, 23(5): 1621--1635, 2012. ISSN 0956-5515. doi: 10.1007/s10845-010-0464-6. URL http://dx.doi.org/10.1007/s10845-010-0464-6.

Robert Bixby, Richard Burda, and D. Miller. Short-interval detailed production scheduling in 300mm semiconductor manufacturing using mixed integer and constraint programming. In *The 17th Annual SEMI/IEEE Advanced Semiconductor Manufacturing Conference (ASMC 2006)*, pages 148--154, 2006. doi: 10.1109/ASMC.2006.1638740. URL http://dx.doi.org/10.1109/ASMC.2006.1638740.

John H. Blackstone, Don T. Phillips, and Gary L. Hogg. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, 20 (1):27--45, 1982.

Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268--308, 2003. ISSN 0360-0300. doi: 10.1145/937503.937505. URL http://doi.acm.org/10.1145/937503.937505.

Christian Blum, Maria José Blesa Aguilera, Andrea Roli, and Michael Sampels, editors. *Hybrid Metaheuristics: An Emerging Approach to Optimization*. Studies in computational intelligence. Springer, Berlin and Heidelberg, 2008. ISBN 9783540782940.

Christian Blum, Jakob Puchinger, Günther R. Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135--4151, 2011. ISSN 1568-4946. doi: 10.1016/j.asoc.2011.02.032. URL http://www.sciencedirect.com/science/article/pii/S1568494611000962.

Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: From natural to artificial systems*. Oxford University Press, New York, 1999. ISBN 0-19-513159-2.

Olivier Bonnin, Dominique Mercier, Didier Levy, Martin Henry, Isabelle Pouilloux, and Eric Mastromatteo. Single-wafer/mini-batch approach for fast cycle time in advanced 300-mm fab. *IEEE Transactions on Semiconductor Manufacturing*, 16(2):111--120, 2003. ISSN 0894-6507. doi: 10.1109/TSM.2003.810920. URL http://dx.doi.org/10.1109/TSM.2003.810920.

Peter C. Bosch and Robert L. Wright. High speed semiconductor fab simulation for large, medium and small lot sizes. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, pages 2305--2312. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6. URL http://dl.acm.org/citation.cfm?id=1516744.1517148.

Mourad Boudhar. Scheduling a batch processing machine with bipartite compatibility graphs. *Mathematical Methods of Operations Research*, 57(3):513--527, 2003. ISSN 1432-2994. doi: 10.1007/s001860300273. URL http://dx.doi.org/10.1007/s001860300273.

Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237(0):82--117, 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2013.02.041. URL http://www.sciencedirect.com/science/article/pii/S0020025513001588.

M. W. Bright, A. R. Hurson, and S. Pakzad. Automated resolution of semantic heterogeneity in multidatabases. *ACM Trans. Database Syst.*, 19(2):212--253, 1994. ISSN 0362-5915. doi: 10.1145/176567.176569. URL http://doi.acm.org/10.1145/176567.176569.

Peter Brucker. *Scheduling algorithms*. Springer, Berlin and New York, 5 edition, 2007. ISBN 978-3-540-69515-8.

Peter Brucker and Sigrid Knust. Complexity results for scheduling problems, 2014. URL http://www.informatik.uni-osnabrueck.de/knust/class/.

Peter Brucker and Mikhail Y. Kovalyov. Single machine batch scheduling to minimize the weighted number of late jobs. *Mathematical Methods of Operations Research*, 43(1):1--8, 1996. ISSN 1432-2994. doi: 10.1007/BF01303431. URL http://dx.doi.org/10.1007/BF01303431.

Peter Brucker, Andrei Gladky, Han Hoogeveen, Mikhail Y. Kovalyov, Chris N. Potts, Thomas Tautenhahn, and Steef L. van de Velde. Scheduling a batching machine. *Journal of Scheduling*, 1(1):31--54, 1998. ISSN 1094-6136.

Mickaël Bureau, Stéphane Dauzère-Pérès, Claude Yugma, Leon Vermariën, and Jean-Bernard Maria. Simulation results and formalism for global-local scheduling in semiconductor manufacturing facilities. In *Proceedings of the 39th Conference on Winter Simulation: 40 Years! The Best is Yet to Come*, WSC '07, pages 1768--1773, Piscataway and NJ and USA, 2007. IEEE Press. ISBN 1-4244-1306-0. URL http://dl.acm.org/citation.cfm?id=1351542.1351859.

Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. A classification of hyper-heuristic approaches. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Stefan Bussmann, Nick Jennings, and Michael J. Wooldridge. *Multiagent systems for manufacturing control: A design methodology*. Springer series on agent technology. Springer, Berlin and London, 2004. ISBN 9783540209249.

Mao-Cheng Cai, Xiaotie Deng, Haodi Feng, Guojun Li, and Guizhen Liu. A ptas for minimizing total completion time of bounded batch scheduling. In William J. Cook and AndreasS Schulz, editors, *Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 304--314. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-43676-8. doi: 10.1007/3-540-47867-1_22. URL http://dx.doi.org/10.1007/3-540-47867-1_22.

Eray Cakici, Scott J. Mason, John W. Fowler, and H. Neil Geismar. Batch scheduling on parallel machines with dynamic job arrivals and incompatible job families. *International Journal of Production Research*, 51(8):2462--2477, 2013. doi: 10.1080/00207543.2012.748227. URL http://dx.doi.org/10.1080/00207543.2012.748227.

E. Campbell and Jim Ammenheuser. 300 mm factory layout and material handling modeling: Phase ii report: Tech transfer document # 99113848b-eng, 2000.

Zhigang Cao and Xiaoguang Yang. A ptas for parallel batch scheduling with rejection and dynamic job arrivals. *Theoretical Computer Science*, 410(27--29):2732--2745, 2009. ISSN 0304-3975. doi: 10.1016/j.tcs.2009.04.006. URL http://www.sciencedirect.com/science/article/pii/S0304397509002990.

Myung Soo Cha, Pyung-Hoi Koo, and Jungjin Joe. Real-time control strategies of batch processing machines in semiconductor manufacturing. In V. Kachitvichyanukul, H.T Luong, and R. Pitakaso, editors, *Proceedings of the Asia Pacific Industrial Engineering & Management Systems Conference 2012*. 2012.

Vijay Chandru and M.R. Rao. Linear programming. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and theory of computation handbook*. CRC Press, Boca Raton [u.a.], 2010. ISBN 978-1-58488-822-2.

Vijaya Chandru, Chung-Yee Lee, and Reha Uzsoy. Minimizing total completion time on batch processing machines. *International Journal of Production Research*, 31(9):2097--2121, 1993a. doi: 10.1080/00207549308956847. URL http://dx.doi.org/10.1080/00207549308956847.

Vijaya Chandru, Chung-Yee Lee, and Reha Uzsoy. Minimizing total completion time on a batch processing machine with job families. *Operations Research Letters*, 13(2):61--65, 1993b. ISSN 0167-6377. doi: 10.1016/0167-6377(93)90030-K. URL http://www.sciencedirect.com/science/article/pii/016763779390030K.

C.-F. Chang and S.-K. Chang. A layer-based layout approach for semiconductor fabrication facilities. In *IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop (ASMC1998)*, pages 385--390, 1998. doi: 10.1109/ASMC.1998.731626. URL http://dx.doi.org/10.1109/ASMC.1998.731626.

Jun-lin Chang, Ying Chen, and Xiao-ping Ma. A hybrid particle swarm optimization algorithm for parallel batch processing machines scheduling. In *Computational Intelligence (UKCI), 2013 13th UK Workshop on*, pages 252--257, 2013. doi: 10.1109/UKCI.2013.6651313. URL http://dx.doi.org/10.1109/UKCI.2013.6651313.

P.-Y. Chang, P. Damodaran, and S. Melouk. Minimizing makespan on parallel batch processing machines. *International Journal of Production Research*, 42(19):4211--4220, 2004. doi: 10.1080/00207540410001711863. URL http://dx.doi.org/10.1080/00207540410001711863.

Pei-Chann Chang and Hui-Mei Wang. A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes. *The International Journal of Advanced Manufacturing Technology*, 24(7-8):615--620, 2004. ISSN 0268-3768. doi: 10.1007/s00170-003-1740-9. URL http://dx.doi.org/10.1007/s00170-003-1740-9.

Bo Chen, Chris N. Potts, and Gerhard J. Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. In Ding-Zhu Du and Panos M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 1493--1641. Springer US, 1999. ISBN 978-1-4613-7987-4. doi: 10.1007/978-1-4613-0303-9_25. URL http://dx.doi.org/10.1007/978-1-4613-0303-9_25.

Bo Chen, Xiaotie Deng, and Wenan Zang. On-line scheduling a batch processing system to minimize total weighted job completion time. In Peter Eades and Tadao Takaoka, editors, *Algorithms and Computation*, volume 2223 of *Lecture Notes in Computer Science*, pages 380--389. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42985-2. doi: 10.1007/3-540-45678-3_33. URL http://dx.doi.org/10.1007/3-540-45678-3_33.

Huaping Chen, Bing Du, and George Q. Huang. Metaheuristics to minimise makespan on parallel batch processing machines with dynamic job arrivals. *International Journal of Computer Integrated Manufacturing*, 23(10):942--956, 2010. doi: 10.1080/0951192X.2010.495137. URL http://dx.doi.org/10.1080/0951192X.2010.495137.

Huaping Chen, Bing Du, and George Q. Huang. Scheduling a batch processing machine with non-identical job sizes: a clustering perspective. *International Journal of Production Research*, 49(19):5755--5778, 2011. doi: 10.1080/00207543.2010.512620. URL http://dx.doi.org/10.1080/00207543.2010.512620.

Jing Chen and Li Jun-qing. Efficient variable neighborhood search for identical parallel machines scheduling. In *Control Conference (CCC), 2012 31st Chinese*, pages 7228--7232, 2012.

Toly Chen. Predicting wafer-lot output time with a hybrid fcm-fbpn approach. *IEEE Transactions on Sytems, Man, and Cybernetics - Part B: Cybernetics*, 37(4):784--793, 2007. doi: 10.1109/TSMCB.2007.895364. URL http://dx.doi.org/10.1109/TSMCB.2007.895364.

Bayi Cheng, Kai Li, and Bo Chen. Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization. *Journal of Manufacturing Systems*, 29(1):29--34, 2010. ISSN 0278-6125. doi: 10.1016/j.jmsy.2010.06.007. URL http://www.sciencedirect.com/science/article/pii/S0278612510000257.

Bayi Cheng, Shan-Lin Yang, Xiaoxuan Hu, and Bo Chen. Minimizing makespan and total completion time for parallel batch processing machines with non-identical job sizes. *Applied Mathematical Modelling*, 36(7):3161--3167, 2012. ISSN 0307-904X. doi: 10.1016/j.apm.2011.09.061. URL http://dx.doi.org/10.1016/j.apm.2011.09.061.

Bayi Cheng, Qi Wang, Shanlin Yang, and Xiaoxuan Hu. An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes. *Applied Soft Computing*, 13(2):765--772, 2013. ISSN 1568-4946. doi: 10.1016/j.asoc.2012.10.021. URL http://www.sciencedirect.com/science/article/pii/S1568494612004735.

T. C. Edwin Cheng, Zhaohui Liu, and Wenci Yu. Scheduling jobs with release dates and deadlines on a batch processing machine. *IIE Transactions*, 33(8):685--690, 2001. ISSN 0740-817X. doi: 10.1023/A:1010991401683. URL http://dx.doi.org/10.1023/A%3A1010991401683.

Tsung-Che Chiang and Li-Chen Fu. Rule-based scheduling in wafer fabrication with due date-based objectives. *Computers & Operations Research*, 39(11):2820--2835, 2012. ISSN 0305-0548. doi: 10.1016/j.cor.2012.02.014. URL http://dx.doi.org/10.1016/j.cor.2012.02.014.

Tsung-Che Chiang, Hsueh-Chien Cheng, and Li-Chen Fu. A memetic algorithm for minimizing total weighted tardiness on parallel batch machines with incompatible job families and dynamic job arrival. *Computers & Operations Research*, 37(12):2257--2269, 2010. ISSN 0305-0548. doi: 10.1016/j.cor.2010.03.017. URL http://dx.doi.org/10.1016/j.cor.2010.03.017.

Chen-Fu Chien, Chih-Wei Hsiao, Meng Cheng, Kuo-Tong Hong, and Szu-Tsung Wang. Cycle time prediction and control based on production line status and manufacturing data mining. In *IEEE International Symposium on Semiconductor Manufacturing 2005 (ISSM 2005)*, pages 327--330, 2005. doi: 10.1109/ISSM.2005.1513369. URL http://dx.doi.org/10.1109/ISSM.2005.1513369.

Chen-Fu Chien, Stéphane Dauzère-Pérès, Hans Ehm, John W. Fowler, Zhibin Jiang, Shekar Krishnaswamy, Lars Mönch, and Reha Uzsoy. Modeling and analysis of semiconductor manufacturing in a shrinking world: challenges and successes. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2093--2099. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6. doi: 10.1109/WSC.2008.4736306. URL http://dx.doi.org/10.1109/WSC.2008.4736306.

Chin Soon Chong, Appa Iyer Sivakumar, and Robert Gay. Simulation based scheduling using a two-pass approach. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, volume 2 of *WSC '03*, pages 1433--1439. ACM Press, 2003a. ISBN 0-7803-8132-7. doi: 10.1109/WSC.2003.1261586. URL http://dx.doi.org/10.1109/WSC.2003.1261586.

Chin Soon Chong, Appa Iyer Sivakumar, and Robert Gay. simulation-based scheduling for dynamic discrete manufacturing. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, WSC '03, pages 1465--1473. ACM Press, 2003b. ISBN 0-7803-8132-7. URL http://dl.acm.org/citation.cfm?id=1030818.1031016.

Fuh-Der Chou and Hui-Mei Wang. Scheduling for a single semiconductor batch-processing machine to minimize total weighted tardiness. *Journal of the Chinese Institute of Industrial Engineers*, 25(2):136--147, 2008. doi: 10.1080/10170660809509079. URL http://dx.doi.org/10.1080/10170660809509079.

Fuh-Der Chou, Pei-Chann Chang, and Hui-Mei Wang. A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 31(3-4):350--359, 2006. ISSN 0268-3768. doi: 10.1007/s00170-005-0194-7. URL http://dx.doi.org/10.1007/s00170-005-0194-7.

Julie Christopher. Study of optimal load lock dedication for cluster tools. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2136--2140. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

Christopher A. Chung. *Simulation modeling handbook: A practical approach*. CRC Press, Boca Raton, 2004. ISBN 0-8493-1241-8.

Jaewoo Chung and Jaejin Jang. The integrated room layout for a semiconductor facility plan. *IEEE Transactions on Semiconductor Manufacturing*, 20(4):517--527, 2007. ISSN 0894-6507. doi: 10.1109/TSM.2007.907621. URL http://dx.doi.org/10.1109/TSM.2007.907621.

Jaewoo Chung and Jaejin Jang. A wip balancing procedure for throughput maximization in semiconductor fabrication. *Semiconductor Manufacturing, IEEE Transactions on*, 22(3):381--390, 2009. ISSN 0894-6507. doi: 10.1109/TSM.2009.2017666. URL http://dx.doi.org/10.1109/TSM.2009.2017666.

S. H. Chung, Y. T. Tai, and W. L. Pearn. Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes. *International Journal of Production Research*, 47(18):5109--5128, 2009. doi: 10.1080/00207540802010807. URL http://dx.doi.org/10.1080/00207540802010807.

R. Cigolini, M. Perona, A. Portioli, and T. Zambelli. A new dynamic look-ahead scheduling procedure for batching machines. *Journal of Scheduling*, 5(2):185--204, 2002. ISSN 1094-6136. doi: 10.1002/jos.99. URL http://dx.doi.org/10.1002/jos.99.

Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1:269--308, 1998a.

Carlos A. Coello Coello. An updated survey of ga-based multiobjective optimization techniques. *ACM Computing Surveys*, 32:109--143, 1998b.

Carlos A. Coello Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In *Proceedings of the Congress on Evolutionary Computation*, pages 3--13. IEEE Press, 1999.

Patrick Cogez, Mart Graef, Bert Huizing, Reinhard Mahnkopf, Hidemi Ishiuchi, Junji Shindo, Siyoung Choi, JaeSung Roh, Carlos H. Diaz, Burn Lin, Ian Steff, Bob Doering, and Paolo Gargini. International technology roadmap for semiconductors: Factory integration, 2007.

Patrick Cogez, Mart Graef, Bert Huizing, Reinhard Mahnkopf, Hidemi Ishiuchi, Junji Shindo, Siyoung Choi, JaeSung Roh, Carlos H. Diaz, Burn Lin, Ian Steff, Bob Doering, and Paolo Gargini. International technology roadmap for semiconductors: Factory integration, 2011.

Paul R. Cohen. *Empirical methods for artificial intelligence*. MIT Press, Cambridge and Mass, 1995. ISBN 0-262-03225-2.

Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151--158, New York and NY and USA, 1971. ACM. doi: 10.1145/800157.805047. URL http://doi.acm.org/10.1145/800157.805047.

Teodor Gabriel Crainic and Michel Toulouse. Parallel meta-heuristics. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Roy Crosbie. Grand challenges in modeling and simulation. In Louis Birta and Azzedine Boukerche, editors, *Proceedings of the Grand Challenges 2010*. Curran Associates, Inc., 2010. ISBN 9781617387043.

S. P. Cunningham and J. George Shanthikumar. Empirical results on the relationship between die yield and cycle time in semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 9(2):273--277, 1996. ISSN 0894-6507. doi: 10.1109/66.492822. URL http://dx.doi.org/10.1109/66.492822.

Rodrigo Ferreira da Silva and Sebastián Urrutia. A general vns heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4):203--211, 2010. ISSN 1572-5286. doi: 10.1016/j.disopt.2010.04.002. URL http://dx.doi.org/10.1016/j.disopt.2010.04.002.

R. M. Dabbas and John W. Fowler. A new scheduling approach using combined dispatching criteria in wafer fabs. *Semiconductor Manufacturing, IEEE Transactions on*, 16(3):501--510, 2003. ISSN 0894-6507. doi: 10.1109/TSM.2003.815201. URL http://dx.doi.org/10.1109/TSM.2003.815201.

P. Damodaran and K. Srihari. Mixed integer formulation to minimize makespan in a flow shop with batch processing machines. *Mathematical and Computer Modelling*, 40(13):1465--1472, 2004. ISSN 0895-7177. doi: 10.1016/j.mcm.2005.01.005. URL http://www.sciencedirect.com/science/article/pii/S0895717705000063.

Purushothaman Damodaran and Ping-Yu Chang. Heuristics to minimize makespan of parallel batch processing machines. *The International Journal of Advanced Manufacturing Technology*, 37(9-10):1005--1013, 2008. ISSN 0268-3768. doi: 10.1007/s00170-007-1042-8. URL http://dx.doi.org/10.1007/s00170-007-1042-8.

Purushothaman Damodaran and Mario C. Vélez-Gallego. Heuristics for makespan minimization on parallel batch processing machines with unequal job ready times. *The International Journal of Advanced Manufacturing Technology*, 49(9-12):1119--1128, 2010. ISSN 0268-3768. doi: 10.1007/s00170-009-2457-1. URL http://dx.doi.org/10.1007/s00170-009-2457-1.

Purushothaman Damodaran and Mario C. Vélez-Gallego. A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times. *Expert Systems with Applications*, 39(1):1451--1458, 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.08.029. URL http://dx.doi.org/10.1016/j.eswa.2011.

Purushothaman Damodaran, Kumar Praveen Manjeshwar, and Krishnaswami Srihari. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. *International Journal of Production Economics*, 103(2):882--891, 2006. ISSN 0925-5273. doi: 10.1016/j.ijpe.2006.02.010. URL http://dx.doi.org/10.1016/j.ijpe.2006.02.010.

Purushothaman Damodaran, Krishnaswami Srihari, and Sarah S. Lam. Scheduling a capacitated batch-processing machine to minimize makespan. *Robotics and Computer-Integrated Manufacturing*, 23(2):208--216, 2007. ISSN 0736-5845. URL http://www.sciencedirect.com/science/article/pii/S0736584506000330.

Purushothaman Damodaran, Mario C. Vélez-Gallego, and Jairo Maya. A grasp approach for makespan minimization on parallel batch processing machines. *Journal of Intelligent Manufacturing*, 22(5):767--777, 2011. ISSN 0956-5515. doi: 10.1007/s10845-009-0272-z. URL http://dx.doi.org/10.1007/s10845-009-0272-z.

Wilhelm Dangelmaier, Kiran R. Mahajan, Thomas Seeger, Benjamin Klöpper, and Mark Aufenanger. Simulation assisted optimization and real-time control aspects of flexible production systems subject to disturbances. In L. Felipe Perrone, Barry Lawson, Jason Liu, and Frederick P. Wieland, editors, *Proceedings of the Winter Simulation Conference*, WSC '06, pages 1785--1795. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7. URL http://dl.acm.org/citation.cfm?id=1218112.1218436.

Wilhelm Dangelmaier, Kiran R. Mahajan, Mark Aufenanger, and Thomas Seeger. Simulation assisted match-up rescheduling of flexible production systems subject to execution exceptions. In Shane G. Henderson, Bahar Biller, Ming-Hua Hsieh, John Shortle, D Tew Jeffrey, and Russell R. Barton, editors, *Proceedings of the 39th Winter Simulation Conference*, WSC '07, pages 1805--1813, Piscataway and NJ and USA, 2007. IEEE Press. ISBN 1-4244-1306-0. URL http://dl.acm.org/citation.cfm?id=1351542.1351866.

Stéphane Dauzère-Pérès and Lars Mönch. Scheduling jobs on a single batch processing machine with incompatible job families and weighted number of tardy jobs objective. *Computers & Operations Research*, 40(5):1224--1233, 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2012.12.012. URL http://www.sciencedirect.com/science/article/pii/S030505481200281X.

Kenneth de Jong, David B. Fogel, and Hans-Paul Schwefel. A history of evolutionary computation. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

Mateus Rocha de Paula, Martín Gómez Ravetti, Geraldo Robson Mateus, and Panos M. Pardalos. Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. *IMA Journal of Management Mathematics*, 18(2):101--115, 2007. doi: 10.1093/imaman/dpm016. URL http://dx.doi.org/10.1093/imaman/dpm016.

Kalyanmoy Deb, Lee Altenberg, Bernard Manderick, Thomas Bäck, Zbigniew Michalewicz, Melanie Mitchell, and Stephanie Forrest. Fitness landscapes. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

Xiaotie Deng, Haodi Feng, Guojun Li, and Guizhen Liu. A ptas for minimizing total completion time of bounded batch scheduling. *International Journal of Foundations of Computer Science*, 13(06):817--827, 2002. doi: 10.1142/S0129054102001473. URL http://dx.doi.org/10.1142/S0129054102001473.

Xiaotie Deng, Chung Poon, and Yuzhong Zhang. Approximation algorithms in batch processing. *Journal of Combinatorial Optimization*, 7(3):247--257, 2003. ISSN 1382-6905. URL http://dx.doi.org/10.1023/A:1027316504440.

Xiaotie Deng, Haodi Feng, Pixing Zhang, Yuzhong Zhang, and Hong Zhu. Minimizing mean completion time in a batch processing system. *Algorithmica*, 38(4):513--528, 2004. ISSN 0178-4617. doi: 10.1007/s00453-003-1053-2. URL http://dx.doi.org/10.1007/s00453-003-1053-2.

A. Devpura, John W. Fowler, M. W. Carlyle, and I. Perez. Minimizing total weighted tardiness on single batch process machine with incompatible job families. In Bernhard Fleischmann, Rainer Lasch, Ulrich Derigs, Wolfgang Domschke, and Ulrich Rieder, editors, *Operations Research Proceedings*, volume 2000 of *Operations Research Proceedings*, pages 366--371. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41587-9. doi: 10.1007/978-3-642-56656-1_58. URL http://dx.doi.org/10.1007/978-3-642-56656-1_58.

Gregory Dobson and Ramakrishnan S. Nambimadom. The batch loading and scheduling problem, 1992.

Gregory Dobson and Ramakrishnan S. Nambimadom. The batch loading and scheduling problem. *Oper. Res.*, 49(1):52--65, 2001. ISSN 0030-364X. doi: 10.1287/opre.49.1.52.11189. URL http://dx.doi.org/10.1287/opre.49.1.52.11189.

Dirk Doleschal. *Vergleich heuristischer und deterministischer Verfahren zur Optimierung von Fertigungsabläufen in der Halbleiterproduktion*. Diploma Thesis, Technische Universität Dresden, Fakultät Mathematik, 2010.

P. D. D. Dominic, S. Kaliyamoorthy, and M. Saravana Kumar. Efficient dispatching rules for dynamic job shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 24(1):70--75, 2004. ISSN 0268-3768. doi: 10.1007/s00170-002-1534-5. URL http://dx.doi.org/10.1007/s00170-002-1534-5.

Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, pages 243--278, 2005. ISSN 0304-3975. doi: 10.1016/j.tcs.2005.05.020. URL http://www.sciencedirect.com/science/article/pii/S0304397505003798.

Marco Dorigo and Thomas Stützle. *Ant colony optimization*. MIT Press, Cambridge and Mass, 2004. ISBN 0-262-04219-3.

Marco Dorigo and Thomas Stützle. Ant colony optimization: Overview and recent advances. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Glenn R. Drake and Jeffrey S. Smith. Simulation system for real-time planning, scheduling, and control. In *Proceedings of the 28th conference on Winter simulation*, WSC '96, pages 1083--1090, Washington and DC and USA, 1996. IEEE Computer Society. ISBN 0-7803-3383-7. doi: 10.1145/256562.256914. URL http://dx.doi.org/10.1145/256562.256914.

Rene Driessel and Lars Mönch. Scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times using variable neighborhood search. In *Proceedings 39th Computers and Industrial Engineering Conference (CIE 2009)*, pages 273--278, 2009. doi: 10.1109/ICCIE.2009.5223515. URL http://dx.doi.org/10.1109/ICCIE.2009.5223515.

Rene Driessel and Lars Mönch. Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. *Computers & Industrial Engineering*, 61(2):336--345, 2011. ISSN 0360-8352. doi: 10.1016/j.cie.2010.07.001. URL http://dx.doi.org/10.1016/j.cie.2010.07.001.

Amine Drira, Henri Pierreval, and Sonia Hajri-Gabouj. Facility layout problems: A survey. *Annual Reviews in Control*, 31(2):255--267, 2007. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2007.04.001. URL http://www.sciencedirect.com/science/article/pii/S1367578807000417.

Ding-Zhu Du and Panos M. Pardalos, editors. *Handbook of combinatorial optimization*. Springer, New York, [online-ausg.] edition, 2005. ISBN 0-387-23830-1.

Izak Duenyas and John J. Neale. Stochastic scheduling of a batch processing machine with incompatible job families. *Annals of Operations Research*, 70(0):191--220, 1997. ISSN 0254-5330. doi: 10.1023/A:1018922104670. URL http://dx.doi.org/10.1023/A%3A1018922104670.

Matthias Dümmler. *Modeling and Optimization of Cluster Tools in Semiconductor Manufacturing*. PhD thesis, Bayerische Julius-Maximilians-Universität Würzburg, Würzburg, 2004.

Lionel Dupont and Clarisse Dhaenens-Flipo. Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure. *Computers & Operations Research*, 29(7):807--819, 2002. ISSN 0305-0548. doi: 10.1016/S0305-0548(00)00078-2. URL http://dx.doi.org/10.1016/S0305-0548(00)00078-2.

J. J. Durillo, A. J. Nebro, F. Luna, Carlos A. Coello Coello, and Enrique Alba. Convergence speed in multi-objective metaheuristics: Efficiency criteria and empirical study. *International Journal for Numerical Methods in Engineering*, 84(11):1344--1375, 2010. ISSN 1097-0207. doi: 10.1002/nme.2944. URL http://dx.doi.org/10.1002/nme.2944.

D. Eberts, R. Rottnick, Germar Schneider, Sophia Keil, Rainer Lasch, and O. Buhmann. Managing variability within wafertest production by combining lean and six sigma. In *23rd Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC2012)*, pages 33--38, 2012. doi: 10.1109/ASMC.2012.6212864. URL http://dx.doi.org/10.1109/ASMC.2012.6212864.

Khaled S. El-Kilany. *Reusable modelling and simulation of flexible manufacturing for next generation semiconductor manufacturing facilities*. PhD thesis, Dublin City University, Dublin, 2004.

Asim Abdel Rahman El Sheikh, Abid Al Ajeeli, and Evon M. O. Abu-Taieh, editors. *Simulation and modeling: Current technologies and applications*. IGI Pub., Hershey, 2008. ISBN 9781599042008.

Bouazza Elbenani, Jacques A. Ferland, and Jonathan Bellemare. Genetic algorithm and large neighbourhood search to solve the cell formation problem. *Expert Systems with Applications*, 39(3):2408--2414, 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.08.089. URL http://www.sciencedirect.com/science/article/pii/S0957417411012206.

Vishnu Erramilli and Scott J. Mason. Multiple orders per job batch scheduling with incompatible jobs. *Annals of Operations Research*, 159(1):245--260, 2008. ISSN 0254-5330. doi: 10.1007/s10479-007-0286-x. URL http://dx.doi.org/10.1007/s10479-007-0286-x.

Larry J. Eshelman. Genetic algorithms. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

D. Fang, J. Hammer, and D. McLeod. The identification and resolution of semantic heterogeneity in multidatabase systems. In *Interoperability in Multidatabase Systems, 1991. IMS '91. Proceedings., First International Workshop on*, pages 136--143, 1991. doi: 10.1109/IMS.1991.153696. URL http://dx.doi.org/10.1109/IMS.1991.153696.

A. Fayed and B. Dunnigan. Characterizing the operating curve - how can semiconductor fabs grade themselves? In *Semiconductor Manufacturing, 2007. ISSM 2007. International Symposium on*, pages 1--4, 2007. doi: 10.1109/ISSM.2007.4446827. URL http://dx.doi.org/10.1109/ISSM.2007.4446827.

Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109--133, 1995. ISSN 0925-5001. doi: 10.1007/BF01096763. URL http://dx.doi.org/10.1007/BF01096763.

J. Ferrell and M. Pratt. I300i factory guidelines: Version 5.0: Technology transfer # 97063311g-eng, 2000.

Paola Festa and Mauricio G. C. Resende. Grasp: An annotated bibliography. *Essays and surveys on metaheuristics*, pages 325--367, 2002.

Paola Festa and Mauricio G. C. Resende. An annotated bibliography of grasp -- part i: Algorithms. *International Transactions in Operational Research*, 16(1):1--24, 2009a. ISSN 1475-3995. doi: 10.1111/j.1475-3995.2009.00663.x. URL http://dx.doi.org/10.1111/j.1475-3995.2009.00663.x.

Paola Festa and Mauricio G. C. Resende. An annotated bibliography of grasp--part ii: Applications. *International Transactions in Operational Research*, 16(2):131--172, 2009b. ISSN 1475-3995. doi: 10.1111/j.1475-3995.2009.00664.x. URL http://dx.doi.org/10.1111/j.1475-3995.2009.00664.x.

Gerd Finke, Vincent Jost, Maurice Queyranne, and András Seb. Batch processing with interval graph compatibilities between tasks. *Discrete Applied Mathematics*, 156(5):556--568, 2008. ISSN 0166-218X. doi: 10.1016/j.dam.2006.03.039. URL http://www.sciencedirect.com/science/article/pii/S0166218X0700114X.

Paul A. Fishwick. *Simulation model design and execution*. Prentice Hall, Englewood Cliffs and N.J, 1995. ISBN 9780130986092.

Paul A. Fishwick and Hyungwook Park. Queue modeling and simulation. In John A. Sokolowski and Catherine M. Banks, editors, *Principles of modeling and simulation*, pages 71--89. John Wiley, Hoboken and N.J, 2009. ISBN 978-0-470-28943-3.

Bobbie L. Foote and Katta G. Murty. Production systems. In A. Ravi Ravindran, editor, *Operations research and management science handbook*. CRC Press, Boca Raton, 2008. ISBN 978-0849397219.

Ken Fordyce, Robert Bixby, and Richard Burda. Technology that upsets the social order - a paradigm shift in assigning lots to tools in a wafer fabricator - the transition from rules to optimization. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2277--2285. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6. doi: 10.1109/WSC.2008.4736331. URL http://dx.doi.org/10.1109/WSC.2008.4736331.

John W. Fowler and Oliver Rose. Grand challenges in modeling and simulation of complex manufacturing systems. *SIMULATION*, 80(9):469--476, 2004. URL http://dblp.uni-trier.de/db/journals/simulation/simulation80.html#FowlerR04.

John W. Fowler, Gary L. Hogg, and Don T. Phillips. Control of multiproduct bulk service diffusion/oxidation processes. *IIE Transactions*, 24(4):84--96, 1992a. ISSN 0740-817X. doi: 10.1080/07408179208964236. URL http://dx.doi.org/10.1080/07408179208964236.

John W. Fowler, Don T. Phillips, and Gary L. Hogg. Real-time control of multiproduct bulk-service semiconductor manufacturing processes. *Semiconductor Manufacturing, IEEE Transactions on*, 5(2):158--163, 1992b. ISSN 0894-6507. doi: 10.1109/66.136278. URL http://dx.doi.org/10.1109/66.136278.

John W. Fowler, Gary L. Hogg, and Don T. Phillips. Control of multiproduct bulk server diffusion/oxidation processes. part 2: multiple servers. *IIE Transactions*, 32(2):167--176, 2000. ISSN 0740-817X. doi: 10.1080/07408170008963889. URL http://dx.doi.org/10.1080/07408170008963889.

John W. Fowler, N. Phojanamongkolkij, Jeffery K. Cochran, and Douglas C. Montgomery. Optimal batching in a wafer fabrication facility using a multiproduct g/g/c model with batch processing. *International Journal of Production Research*, 40(2):275--292, 2002. URL http://www.ingentaconnect.com/content/tandf/tprs/2002/00000040/00000002/art00002.

John W. Fowler, Michele E. Pfund, Scott J. Mason, Oliver Rose, Lars Mönch, and Roland Sturm. Deterministic scheduling of wafer fab operations. In Brooks Automation Inc., editor, *Worldwide Automation Symposium 2003*. Brooks Automation Inc., Phoenix and AZ and USA, 2003. ISBN 0-9729100-2-6.

Alexey Frantsuzov. *Automated Analytical Equipment Modeling in Semiconductor Manufacturing*. Master Thesis, Technische Universität Dresden, Faculty of Computer Science, Dresden, 2011.

Lawrence D. Fredendall, Divesh Ojha, and J. Wayne Patterson. Concerning the theory of workload control. *European Journal of Operational Research*, 201(1):99--111, 2010. ISSN 0377-2217. doi: 10.1016/j.ejor.2009.02.003. URL http://dx.doi.org/10.1016/j.ejor.2009.02.003.

Bernd Freisleben. Metaevolutionary approaches. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

Michael C. Fu, Fred Glover, and J. April. Simulation optimization: a review, new developments, and applications. In *Proceedings of the Winter Simulation Conference 2005*, page 13 pp, 2005. doi: 10.1109/WSC.2005.1574242. URL http://dx.doi.org/10.1109/WSC.2005.1574242.

V. K. Ganesan, Amit Kumar Gupta, and S. A. Iyer. Bi-objective schedule control of batch processes in semiconductor manufacturing. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 2, pages 1077--1082 vol.2, 2004. doi: 10.1109/RAMECH.2004.1438069. URL http://dx.doi.org/10.1109/RAMECH.2004.1438069.

M. R. Garey and D. S. Johnson. Strong np-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499--508, 1978. ISSN 0004-5411. doi: 10.1145/322077.322090. URL http://doi.acm.org/10.1145/322077.322090.

Natarajan Gautam. Queueing theory. In A. Ravi Ravindran, editor, *Operations research and management science handbook*. CRC Press, Boca Raton, 2008. ISBN 978-0849397219.

Michel Gendreau and Jean-Yves Potvin. Tabu search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Ashish Ghosh and Satchidananda Dehuri. Evolutionary algorithms for multi-criterion optimization: A survey. In *International Journal of Computing & Information Sciences*, volume Vol 2, No. 1, pages 38--57. 2004.

Jay B. Ghosh and Jatinder N. D. Gupta. Batch scheduling to minimize maximum lateness. *Operations Research Letters*, 21(2):77--80, 1997. ISSN 0167-6377. doi: 10.1016/S0167-6377(97)00028-X. URL http://www.sciencedirect.com/science/article/pii/S016763779700028X.

Mike Gißrau. *Development and Introduction of a Semiconductor Production Control Strategy in Terms of Customer Oriented Manufacturing*. PhD thesis, Universität der Bundeswehr München, Neubiberg, 2013.

C. R. Glassey and W. W. Weng. Dynamic batching heuristic for simultaneous processing. *Semiconductor Manufacturing, IEEE Transactions on*, 4(2):77--82, 1991. ISSN 0894-6507. doi: 10.1109/66.79719. URL http://dx.doi.org/10.1109/66.79719.

Moacir Godinho Filho and Reha Uzsoy. The effect of shop floor continuous improvement programs on the lot size--cycle time relationship in a multi-product single-machine environment. *The International Journal of Advanced Manufacturing Technology*, 52(5-8):669--681, 2011. ISSN 0268-3768. doi: 10.1007/s00170-010-2770-8. URL http://dx.doi.org/10.1007/s00170-010-2770-8.

Ravindra Gokhale and M. Mathirajan. Heuristic algorithms for scheduling of a batch processor in automobile gear manufacturing. *International Journal of Production Research*, 49 (10):2705--2728, 2011. doi: 10.1080/00207541003720368. URL http://dx.doi.org/10.1080/00207541003720368.

Ravindra Gokhale and M. Mathirajan. Minimizing total weighted tardiness on heterogeneous batch processors with incompatible job families. *The International Journal of Advanced Manufacturing Technology*, 70(9-12):1563--1578, 2014. ISSN 0268-3768. doi: 10.1007/s00170-013-5324-z. URL http://dx.doi.org/10.1007/s00170-013-5324-z.

Aitor Goti. *Discrete event simulations*. Sciyo, Rijeka, 2010. ISBN 978-953-307-115-2.

Manish K. Govil and Michael C. Fu. Queueing theory in manufacturing: A survey. *Journal of Manufacturing Systems*, 18(3):214--240, 1999. ISSN 0278-6125. doi: 10.1016/S0278-6125(99)80033-8. URL http://www.sciencedirect.com/science/article/pii/S0278612599800338.

R. L. Graham, Eugene L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In Peter L. Hammer, E.L Johnson, and B. H. Korte, editors, *Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*, volume 5 of *Annals of Discrete Mathematics*, pages 287--326. Elsevier Science, 1979. doi: 10.1016/S0167-5060(08)70356-X. URL http://dx.doi.org/10.1016/S0167-5060(08)70356-X.

H. J. Greenberg and Tod Morrison. Robust optimization. In A. Ravi Ravindran, editor, *Operations research and management science handbook*. CRC Press, Boca Raton, 2008. ISBN 978-0849397219.

Julie Greensmith, Amanda Whitbrook, and Uwe Aickelin. Artificial immune systems. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Chengtao Guo, Zhibin Jiang, and Hongtao Hu. A hybrid ant colony optimization method for scheduling batch processing machine in the semiconductor manufacturing. In *Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on*, pages 1698--1701, 2010. doi: 10.1109/IEEM.2010.5674588. URL http://dx.doi.org/10.1109/IEEM.2010.5674588.

Amit Kumar Gupta and Appa Iyer Sivakumar. Simulation based multiobjective schedule optimization in semiconductor manufacturing. In Enver Yücesan, C. H. Chen, Jane L. Snowdon, and J. M. Charnes, editors, *Proceedings of the 34th Winter simulation Conference*, volume 2 of *WSC '02*, pages 1862 -- 1870 vol.2. Winter Simulation Conference, 2002. ISBN 0-7803-7615-3. doi: 10.1109/WSC.2002.1166480. URL http://dx.doi.org/10.1109/WSC.2002.1166480.

Amit Kumar Gupta and Appa Iyer Sivakumar. Optimization of due-date objectives in scheduling semiconductor batch manufacturing. *International Journal of Machine Tools and Manufacture*, 46(12--13):1671--1679, 2006. ISSN 0890-6955. doi: 10.1016/j.ijmachtools.2005.08.017. URL http://dx.doi.org/10.1016/j.ijmachtools.2005.08.017.

Amit Kumar Gupta, S. A. Iyer, and V. K. Ganesan. Look ahead batching to minimize earliness/tardiness measures in batch processes. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 2, pages 1101--1106 vol.2, 2004. doi: 10.1109/RAMECH.2004.1438073. URL http://dx.doi.org/10.1109/RAMECH.2004.1438073.

Amit Kumar Gupta, Peter Lendermann, Appa Iyer Sivakumar, and John Priyadi. Simulation analysis of cluster tool operations in wafer fabrication. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2141--2147. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

Jatinder N. D. Gupta, R. Ruiz, John W. Fowler, and Scott J. Mason. Operational planning and control of semiconductor wafer production. *Production Planning & Control*, 17(7):639--647, 2006. doi: 10.1080/09537280600900733. URL http://dx.doi.org/10.1080/09537280600900733.

H. Gurnani, R. Anupindi, and R. Akella. Control of batch processing systems. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1772--1777 vol.2, 1991. doi: 10.1109/ROBOT.1991.131879. URL http://dx.doi.org/10.1109/ROBOT.1991.131879.

H. Gurnani, R. Anupindi, and R. Akella. Control of batch processing systems in semiconductor wafer fabrication facilities. *Semiconductor Manufacturing, IEEE Transactions on*, 5(4):319--328, 1992. ISSN 0894-6507. doi: 10.1109/66.175364. URL http://dx.doi.org/10.1109/66.175364.

Ilka Habenicht and Lars Mönch. Simulation-based assessment of batching heuristics in semiconductor manufacturing. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, volume 2 of *WSC '03*, pages 1338--1345. ACM Press, 2003. ISBN 0-7803-8132-7. doi: 10.1109/WSC.2003.1261570. URL http://dx.doi.org/10.1109/WSC.2003.1261570.

Gideon Halevi. *Handbook of production management methods*. Butterworth-Heinemann, Oxford and UK, 2001. ISBN 0-7506-5088-5.

Myoungsoo Ham and John W. Fowler. Scheduling of wet etch and furnace operations with next arrival control heuristic. *The International Journal of Advanced Manufacturing Technology*, 38 (9):1006--1017, 2008. ISSN 0268-3768. doi: 10.1007/s00170-007-1146-1. URL http://dx.doi.org/10.1007/s00170-007-1146-1.

Myoungsoo Ham, Young-Hoon Lee, and John W. Fowler. Integer programming-based real-time scheduler in semiconductor manufacturing. In *Winter Simulation Conference*, WSC '09, pages 1657--1666. Winter Simulation Conference, 2009. ISBN 978-1-4244-5771-7. URL http://dl.acm.org/citation.cfm?id=1995456.1995681.

Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449--467, 2001. ISSN 0377-2217. doi: 10.1016/S0377-2217(00)00100-4. URL http://dx.doi.org/10.1016/S0377-2217(00)00100-4.

Pierre Hansen and Nenad Mladenović. A tutorial on variable neighborhood search, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=10229A66CF7FB41C7664B5E139504E41?doi=10.1.1.4.2350&rep=rep1&type=pdf.

Pierre Hansen, Nenad Mladenović, and Dionisio Perez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335--350, 2001. ISSN 1381-1231. URL http://dx.doi.org/10.1023/A:1011336210885.

Pierre Hansen, Nenad Mladenović, Jack Brimberg, and Jose A. Moreno Perez. Variable neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

R. Hase, C. G. Takoudis, and Reha Uzsoy. Cellular and reentrant layouts for semiconductor wafer fabrication facilities. In *Proceedings of the Sixteenth IEEE/CPMT International Electronics Manufacturing Technology Symposium (IEMT1994)*, pages 112 --118 vol.1, 1994. doi: 10.1109/IEMT.1994.404681. URL http://dx.doi.org/10.1109/IEMT.1994.404681.

R. Hase, Reha Uzsoy, and C. G. Takoudis. Alternative facility layouts for semiconductor wafer fabrication facilities. In *Seventeenth IEEE/CPMT International Electronics Manufacturing Technology Symposium*, pages 384--388, 1997. doi: 10.1109/IEMT.1995.526191. URL http://dx.doi.org/10.1109/IEMT.1995.526191.

R. Haupt. A survey of priority rule-based scheduling. *OR Spectrum*, 11(1):3--16, 1989. doi: 10.1007/BF01721162. URL http://dx.doi.org/10.1007/BF01721162.

Shawn Hedman. *A first course in logic: An introduction to model theory, proof theory, computability, and complexity*. Oxford University Press, Oxford and and New York, 2004. ISBN 0-19-852980-5.

Harald Heinrich and Arthur Deutschländer. Rules of automation for 200mm semiconductor manufacturing environment. In *23rd Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC2012)*, pages 51--56, 2012. doi: 10.1109/ASMC.2012.6212867. URL http://dx.doi.org/10.1109/ASMC.2012.6212867.

Harald Heinrich, Germar Schneider, Frank Heinlein, Sophia Keil, Arthur Deutschländer, and Rainer Lasch. Pursuing the increase of factory automation in 200mm frontend manufacturing to manage the changes imposed by the transition from high-volume low-mix to high-mix low-volume production. In *IEEE/SEMI Advanced Semiconductor Manufacturing Conference 2008 (ASMC2008)*, pages 148--155, 2008. doi: 10.1109/ASMC.2008.4529020. URL http://dx.doi.org/10.1109/ASMC.2008.4529020.

Frederick S. Hillier and Gerald J. Lieberman. *Introduction to operations research*. McGraw-Hill, Boston, 7 edition, 2001. ISBN 0-07-232169-5.

Yu-Chi Ho and D.L Pepyne. Simple explanation of the no free lunch theorem of optimization. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on, title=*, volume 5, pages 4409 --4414 vol.5, 2001. doi: 10.1109/.2001.980896. URL http://dx.doi.org/10.1109/.2001.980896.

Dorit S. Hochbaum and Dan Landy. Scheduling with batching: minimizing the weighted number of tardy jobs. *Operations Research Letters*, 16(2):79--86, 1994. ISSN 0167-6377. doi: 10.1016/0167-6377(94)90063-9. URL http://www.sciencedirect.com/science/article/pii/0167637794900639.

Dorit S. Hochbaum and Dan Landy. Scheduling semiconductor burn-in operations to minimize total flowtime. *Operations Research*, 45(6):874, 1997. URL http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=189626&site=ehost-live.

D. J. Hoitomt and P. B. Luh. Scheduling a batch processing facility. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1167--1172 vol.2, 1992. doi: 10.1109/ROBOT.1992.220091. URL http://dx.doi.org/10.1109/ROBOT.1992.220091.

Oliver Holthaus and Chandrasekharan Rajendran. Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, 48(1):87--105, 1997. ISSN 0925-5273. doi: 10.1016/S0925-5273(96)00068-0. URL http://dx.doi.org/10.1016/S0925-5273(96)00068-0.

J. N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33--42, 1995. ISSN 1381-1231. doi: 10.1007/BF02430364. URL http://dx.doi.org/10.1007/BF02430364.

Wallace J. Hopp and Mark L. Spearman. *Factory physics: Foundations of manufacturing management*. Irwin/McGraw-Hill, Boston, 2 edition, 2001. ISBN 0256247951.

Sven Horn. *Simulationsgestützte Optimierung von Fertigungsabläufen in der Produktion elektronischer Halbleiterspeicher*. Detert, Templin, 1 edition, 2008. ISBN 3934142303.

Sven Horn, Gerald Weigert, and Sebastian Werner. Data coupling strategies in production environments. In *28th International Spring Seminar on Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 2005.*, pages 278--282, 2005. doi: 10.1109/ISSE.2005.1491040. URL http://dx.doi.org/10.1109/ISSE.2005.1491040.

Sven Horn, Gerald Weigert, Sebastian Werner, and Thomas Jähnig. Simulation based scheduling system in a semiconductor backend facility. In L. Felipe Perrone, Barry Lawson, Jason Liu, and Frederick P. Wieland, editors, *Proceedings of the Winter Simulation Conference*, WSC '06, pages 1741--1748. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7. URL http://dl.acm.org/citation.cfm?id=1218112.1218428.

Hisashi Hosoe, Nobuo Knanamori, and Kouhei Yoshida. The methods of data collection and tool processing time estimation in lot processing. In *Proceedings of the International Symposium on Semiconductor Manufacturing (ISSM 2007)*, pages 1--4, 2007. doi: 10.1109/ISSM.2007.4446810. URL http://dx.doi.org/10.1109/ISSM.2007.4446810.

B. Hrúz and MengChu Zhou. *Modeling and control of discrete-event dynamical systems: With Petri nets and other tool*. Springer, London, 2007. ISBN 9781846288722.

Bin Hu and Günther R. Raidl. Variable neighborhood descent with selfadaptive neighborhood-ordering. In *Proceedings of the 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics*, 2006.

Bin Hu and Günther R. Raidl. Effective neighborhood structures for the generalized traveling salesman problem. In Jano van Hemert and Carlos Cotta, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 4972 of *Lecture Notes in Computer Science*, pages 36--47. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78603-0. URL http://dx.doi.org/10.1007/978-3-540-78604-7_4.

Yi-Feng Hung and Ching-Bin Chang. Dispatching rules using flow time predictions for semiconductor wafer fabrications. *Journal of the Chinese Institute of Industrial Engineers*, 19(1):67--75, 2002. doi: 10.1080/10170660209509184. URL http://dx.doi.org/10.1080/10170660209509184.

Toshihide Ibaraki. Comparison with existing optimization methods. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

Yoshiro Ikura and Mark Gimple. Efficient scheduling algorithms for a single batch processing machine. *Operations Research Letters*, 5(2):61--65, 1986. ISSN 0167-6377. doi: 10.1016/0167-6377(86)90104-5. URL http://www.sciencedirect.com/science/article/pii/0167637786901045.

ISO 14644-1. Cleanrooms and associated controlled environments---part 1: Classification of air cleanliness, 1999. URL http://www.iest.org/StandardsRPs/ISOStandards/ISO14644Standards/ISO1464411999/tabid/10136/Default.aspx.

ISO/IEC 15288. Iso/iec/ieee systems and software engineering - system life cycle processes: System life cycle processes, 2008.

Nicholas R. Jennings and Michael J. Wooldridge. *Agent technology: Foundations, applications, and markets*. Springer-Verlag, Berlin [etc.], 1998. ISBN 978-3540635918.

Abdessalem Jerbi and Hédi Chtourou. Cellular or functional layout? In Vladimir Modrak and R. Sudhakara Pandian, editors, *Operations management research and cellular manufacturing systems*, pages 189--206. Business Science Reference, Hershey PA, 2012. ISBN 978-1-61350-048-4.

Wenyou Jia, Zhibin Jiang, and You Li. A job-family-oriented algorithm for re-entrant batch processing machine scheduling. In *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, pages 1022--1027, 2013. doi: 10.1109/CoASE.2013.6653907. URL http://dx.doi.org/10.1109/CoASE.2013.6653907.

Zhao-hong Jia and Joseph Y-T. Leung. An improved meta-heuristic for makespan minimization of a single batch machine with non-identical job sizes. *Computers & Operations Research*, 46(0):49--58, 2014. ISSN 0305-0548. URL http://www.sciencedirect.com/science/article/pii/S0305054814000021.

Tao Jiang, Li Ming, Bala Ravikumar, and Kenneth W. Regan. Computability. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and theory of computation handbook*. CRC Press, Boca Raton [u.a.], 2010. ISBN 978-1-58488-822-2.

Gao Jie, Sun Linyan, and Gen Mitsuo. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9): 2892--2907, 2008. ISSN 0305-0548. doi: 10.1016/j.cor.2007.01.001. URL http://dx.doi.org/10.1016/j.cor.2007.01.001.

Carl Johnzén, Stéphane Dauzère-Pérès, P. Vialletelle, and C. Yugma. Importance of qualification management for wafer fabs. In *Advanced Semiconductor Manufacturing Conference, 2007. ASMC 2007. IEEE/SEMI*, pages 166--169, 2007. doi: 10.1109/ASMC.2007.375107. URL http://dx.doi.org/10.1109/ASMC.2007.375107.

Carl Johnzén, Philippe Vialletelle, Stéphane Dauzère-Pérès, Claude Yugma, and Alexandre Derreumaux. Impact of qualification management on scheduling in semiconductor manufacturing. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, pages 2059--2066. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6. URL http://dl.acm.org/citation.cfm?id=1516744.1517103.

Carl Johnzén, Stéphane Dauzère-Pérès, and Philippe Vialletelle. Flexibility measures for qualification management in wafer fabs. *Production Planning & Control*, 22(1):81--90, 2011. doi: 10.1080/09537287.2010.490022. URL http://dx.doi.org/10.1080/09537287.2010.490022.

Pravin K. Johri. Practical issues in scheduling and dispatching in semiconductor wafer fabrication. *Journal of Manufacturing Systems*, 12(6):474--485, 1993. ISSN 0278-6125. doi: 10.1016/0278-6125(93)90344-S. URL http://www.sciencedirect.com/science/article/pii/027861259390344S.

Fariborz Jolai. Minimizing number of tardy jobs on a batch processing machine with incompatible job families. *European Journal of Operational Research*, 162(1):184--190, 2005. ISSN 0377-2217. doi: 10.1016/j.ejor.2003.10.011. URL http://www.sciencedirect.com/science/article/pii/S0377221703007392.

Fariborz Jolai Ghazvini and Lionel Dupont. Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes. *International Journal of Production Economics*, 55(3):273--280, 1998. ISSN 0925-5273. URL http://www.sciencedirect.com/science/article/pii/S092552739800067X.

Li Jun-Qing, Pan Quan-Ke, and Xie Sheng-Xian. A hybrid variable neighborhood search algorithm for solving multi-objective flexible job shop problems. *Computer Science and Information Systems*, 7(4):907--930, 2010.

Chihyun Jung and Tae-Eog Lee. Empirical results on the relationship between die yield and cycle time in semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 25(2):186--199, 2012. ISSN 0894-6507.

John J. Kanet, Sanjay L. Ahire, and Michael F. Gorman. Constraint programming for scheduling. In Joseph Y-T. Leung, editor, *Handbook of scheduling*. Chapman & Hall/CRC, Boca Raton, 2004. ISBN 9780203489802.

R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85--103. Plenum Press, 1972.

Ali Husseinzadeh Kashan and Behrooz Karimi. Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: An ant colony framework. *Journal of the Operational Research Society*, 59(9):1269--1280, 2007. doi: 10.1057/palgrave.jors.2602448. URL http://dx.doi.org/10.1057/palgrave.jors.2602448.

Ali Husseinzadeh Kashan, Behrooz Karimi, and Fariborz Jolai. Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes. *International Journal of Production Research*, 44(12):2337--2360, 2006a. doi: 10.1080/00207540500525254. URL http://dx.doi.org/10.1080/00207540500525254.

Ali Husseinzadeh Kashan, Behrooz Karimi, and Fariborz Jolai. Minimizing makespan on a single batch processing machine with non-identical job sizes: A hybrid genetic approach. In Jens Gottlieb and Günther R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 135--146. Springer Berlin Heidelberg, 2006b. ISBN 978-3-540-33178-0. doi: 10.1007/11730095_12. URL http://dx.doi.org/10.1007/11730095_12.

Ali Husseinzadeh Kashan, Behrooz Karimi, and Masoud Jenabi. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Computers & Operations Research*, 35(4):1084--1098, 2008. ISSN 0305-0548. doi: 10.1016/j.cor.2006.07.005. URL http://dx.doi.org/10.1016/j.cor.2006.07.005.

Ali Husseinzadeh Kashan, Behrooz Karimi, and S.M.T. Fatemi Ghomi. A note on minimizing makespan on a single batch processing machine with nonidentical job sizes. *Theoretical Computer Science*, 410(27--29):2754--2758, 2009. ISSN 0304-3975. doi: 10.1016/j.tcs.2009.02.014. URL http://dx.doi.org/10.1016/j.tcs.2009.02.014.

Ali Husseinzadeh Kashan, Behrooz Karimi, and Fariborz Jolai. An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes. *Engineering Applications of Artificial Intelligence*, 23(6):911--922, 2010. ISSN 0952-1976. doi: 10.1016/j.engappai.2010.01.031. URL http://www.sciencedirect.com/science/article/pii/S0952197610000722.

Sophia Keil, D. Eberts, Thomas Igel, Germar Schneider, Kristina Wilhelm, Rainer Lasch, and Arthur Deutschländer. Innovation and manufacturing excellence in mature multi-product semiconductor fabrication facilities via design for flow by 3. In *Semiconductor Conference Dresden (SCD2011)*, pages 1--5, 2011. doi: 10.1109/SCD.2011.6068762. URL http://dx.doi.org/10.1109/SCD.2011.6068762.

Karl G. Kempf, Reha Uzsoy, and Cheng-Shuo Wang. Scheduling a single batch processing machine with secondary resource constraints. *Journal of Manufacturing Systems*, 17(1):37--51, 1998. ISSN 0278-6125. doi: 10.1016/S0278-6125(98)80008-3. URL http://www.sciencedirect.com/science/article/pii/S0278612598800083.

Katariina Kemppainen. *Priority scheduling revisited: Dominant rules, open protocols, and integrated order management*. Helsinki School of Economics, Helsinki, 2005. ISBN 9517919689.

Mohammad Khajehzadeh, Mohd Raihan Taha, Ahmed El-Shafie, and Mahdiyeh Eslami. A survey on meta-heuristic global optimization algorithms. In *Research Journal of Applied Sciences, Engineering and Technology*, volume 3, pages 569--578. Maxwell Science Publication, 2011.

Yeong-Dae Kim, Dong-Ho Lee, Jung-Ug Kim, and Hwan-Kyun Roh. A simulation study on lot release control, mask scheduling, and batch scheduling in semiconductor wafer fabrication facilities. *Journal of Manufacturing Systems*, 17(2):107--117, 1998. ISSN 0278-6125. doi: 10.1016/S0278-6125(98)80024-1. URL http://www.sciencedirect.com/science/article/pii/S0278612598800241.

Yeong-Dae Kim, Jae-Gon Kim, Bum Choi, and Hyung-Un Kim. Production scheduling in a semiconductor wafer fabrication facility producing multiple product types with distinct due dates. *Robotics and Automation, IEEE Transactions on*, 17(5):589--598, 2001. ISSN 1042-296X. doi: 10.1109/70.964660. URL http://dx.doi.org/10.1109/70.964660.

Yeong-Dae Kim, Byung-Jun Joo, and So-Young Choi. Scheduling wafer lots on diffusion machines in a semiconductor wafer fabrication facility. *Semiconductor Manufacturing, IEEE Transactions on*, 23(2):246--254, 2010. ISSN 0894-6507. doi: 10.1109/TSM.2010.2045666. URL http://dx.doi.org/10.1109/TSM.2010.2045666.

Kenneth E. JR Kinnear, Robert E. Smith, and Zbigniew Michalewicz. Derivative methods: (ec). In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

Gokhan Kirlik and Ceyda Oguz. A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. *Computers & Operations Research*, 39(7):1506--1520, 2012. ISSN 0305-0548. doi: 10.1016/j.cor.2011.08.022. URL http://dx.doi.org/10.1016/j.cor.2011.08.022.

Philip N. Klein and Neal E. Young. Approximation algorithms for np-hard optimization problems. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and theory of computation handbook*. CRC Press, Boca Raton [u.a.], 2010. ISBN 978-1-58488-822-2.

Andreas Klemmt. *Ablaufplanung in der Halbleiter- und Elektronikproduktion: Hybride Optimierungsverfahren und Dekompositionstechniken*. Vieweg+Teubner Verlag, Wiesbaden, 2012. ISBN 9783834819949.

Andreas Klemmt and Lars Mönch. Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing. In *Proceedings of the Winter Simulation Conference*, WSC '12, pages 194:1--194:10. Winter Simulation Conference, 2012. URL http://dl.acm.org/citation.cfm?id=2429759.2430019.

Andreas Klemmt, Sven Horn, Gerald Weigert, and T. Hielscher. Simulations-based and solver-based optimization approaches for batch processes in semiconductor manufacturing. In *Simulation Conference, 2008. WSC 2008. Winter*, pages 2041--2049, 2008. doi: 10.1109/WSC.2008.4736300. URL http://dx.doi.org/10.1109/WSC.2008.4736300.

Andreas Klemmt, Gerald Weigert, Christian Almeder, and Lars Mönch. A comparison of mip-based decomposition techniques and vns approaches for batch scheduling problems. In M. D. Rossetti, Raymond R. Hill, Björn Johansson, Ann Dunkin, and Ricki G. Ingalls, editors, *Proceedings of the Winter Simulation Conference*, WSC '09, pages 1686--1694, 2009. doi: 10.1109/WSC.2009.5429173. URL http://dx.doi.org/10.1109/WSC.2009.5429173.

Andreas Klemmt, Gerald Weigert, and Sebastian Werner. Optimisation approaches for batch scheduling in semiconductor manufacturing. *European Journal of Industrial Engineering*, 5(3): 338--359, 2011.

Hyo-Heon Ko, Jihyun Kim, Sung-Shick Kim, and Jun-Geol Baek. Dispatching rule for non-identical parallel machines with sequence-dependent setups and quality restrictions. *Computers & Industrial Engineering*, 59(3):448--457, 2010. ISSN 0360-8352. doi: 10.1016/j.cie.2010.05.017. URL http://dx.doi.org/10.1016/j.cie.2010.05.017.

Eric J. Koehler, Timbur M. Wulf, Alvin C. Bruska, and Marvin S. Seppanen. Evaluation of cluster tool throughput for thin film head production. In P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, editors, *Proceedings of the 31st Winter Simulation Conference*, WSC '99, pages 714--719, New York, 1999. ACM Press. ISBN 0-7803-5780-9.

Frederic Koehler and Samir Khuller. Optimal batch schedules for parallel machines. In Frank Dehne, Roberto Solis-Oba, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, volume 8037 of *Lecture Notes in Computer Science*, pages 475--486. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40103-9. doi: 10.1007/978-3-642-40104-6_41. URL http://dx.doi.org/10.1007/978-3-642-40104-6_41.

Shie-Gheun Koh, Pyung-Hoi Koo, Jae-Won Ha, and Woon-Seek Lee. Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job families. *International Journal of Production Research*, 42(19):4091--4107, 2004. doi: 10.1080/00207540410001704041. URL http://dx.doi.org/10.1080/00207540410001704041.

Shie-Gheun Koh, Pyung-Hoi Koo, Dong-Chun Kim, and Won-Suk Hur. Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. *International Journal of Production Economics*, 98(1):81--96, 2005. ISSN 0925-5273. doi: 10.1016/j.ijpe.2004.10.001. URL http://dx.doi.org/10.1016/j.ijpe.2004.10.001.

Robert Kohn and Oliver Rose. Automated generation of analytical process time models for cluster tools in semiconductor manufacturing. In S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and Michael C. Fu, editors, *Proceedings of the Winter Simulation Conference*, pages 1803--1815, 2011. doi: 10.1109/WSC.2011.6147895. URL http://dx.doi.org/10.1109/WSC.2011.6147895.

Robert Kohn and Oliver Rose. Study on optimization potential influencing factors in simulation studies focused on parallel batch machine scheduling using variable neighbourhood search. In *Simulation Conference (WSC), Proceedings of the 2012 Winter*, pages 1--13, 2012. doi: 10.1109/WSC.2012.6465273. URL http://dx.doi.org/10.1109/WSC.2012.6465273.

Robert Kohn and Oliver Rose. The impact of accuracy in lot arrival prediction on solution quality for the parallel batch machine scheduling problem in wafer fabrication. In *ASIM Dedicated Conferences on Simulation in Production and Logistics*. 2013.

Robert Kohn, Oliver Rose, and Sebastian Werner. Automated semiconductor equipment modeling and model parameter estimation using mes data. In *Proceedings of the IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 11--16, 2010. doi: 10.1109/ASMC.2010.5551413. URL http://dx.doi.org/10.1109/ASMC.2010.5551413.

Robert Kohn, Oliver Rose, and Christoph Laroque. Study on multi-objective optimization for parallel batch machine scheduling using variable neighbourhood search. In *Simulation Conference (WSC), 2013 Winter*, pages 3654--3670, 2013. doi: 10.1109/WSC.2013.6721726. URL http://dx.doi.org/10.1109/WSC.2013.6721726.

Gediz Korkmaz. Batch scheduling of incompatible jobs on a single reactor with dynamic arrivals: Master thesis, 2004.

B. H. Korte and Jens Vygen. *Combinatorial optimization: Theory and algorithms*. Springer Berlin Heidelberg, Heidelberg and and New York, 5 edition, 2012. ISBN 978-3-642-24488-9.

Annette J. Krygiel. *Behind the Wizard's curtain: An integration environment for a system of systems*. National Defense University, 1999. ISBN 1579060188.

Ajay D. Kshemkalyani and Mukesh Singhal. *Distributed computing: Principles, algorithms, and systems*. Cambridge University Press, Cambridge, 2008. ISBN 978-0-521-87634-6.

Sameer Kumar and Daniel A. Nottestad. Integrated simulation application design for short-term production scheduling. *IIE Transactions*, 38(9):737--748, 2006. ISSN 0740-817X. URL http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=21524195&site=ehost-live.

Sunil Kumar and P. R. Kumar. Queueing network models in the design and analysis of semiconductor wafer fabs. *IEEE Transactions on Robotics and Automation*, 17:548--561, 2000.

Mary E. Kurz and Scott J. Mason. Minimizing total weighted tardiness on a batch-processing machine with incompatible job families and job ready times. *International Journal of Production Research*, 46(1):131--151, 2008. URL http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=27529851&site=ehost-live.

Miros-law Lachowicz and Jacek Miekisz, editors. *From genetics to mathematics*. World Scientific Pub. Co., Singapore and and Hackensack and N.J, 2009. ISBN 978-981-283-724-0.

B. J. Lageweg, J. K. Lenstra, Eugene L. Lawler, and A. H. G. Rinnooy Kan. Computer-aided complexity classification of combinational problems. *Commun. ACM*, 25(11):817--822, 1982. ISSN 0001-0782. doi: 10.1145/358690.363066. URL http://doi.acm.org/10.1145/358690.363066.

William B. Langdon, Robert I. McKay, and Lee Spector. Genetic programming. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Jan Lange. Automated generation and parameterization of throughput models for semiconductor tools: Diploma thesis, 2008.

Jan Lange, Kilian Schmidt, Roy Borner, and Oliver Rose. Automated generation and parameterization of throughput models for semiconductor tools. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2335--2340. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

Jan Lange, Andreas Klemmt, and Gerald Weigert. Generic visualization of technological process flows. In *Electronics Technology, 2009. ISSE 2009. 32nd International Spring Seminar on*, pages 1--5, 2009. doi: 10.1109/ISSE.2009.5206935. URL http://dx.doi.org/10.1109/ISSE.2009.5206935.

Averill M. Law and W. David Kelton. *Simulation modeling and analysis*. McGraw-Hill, Boston, 3 edition, 2000. ISBN 0070592926.

Eugene L. Lawler. A ''pseudopolynomial'' algorithm for sequencing jobs to minimize total tardiness. In Peter L. Hammer, E.L Johnson, B. H. Korte, and G. L. Nemhauser, editors, *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 331--342. Elsevier Science, 1977. doi: 10.1016/S0167-5060(08)70742-8. URL http://dx.doi.org/10.1016/S0167-5060(08)70742-8.

Robert C. Leachman. *Competitive Semiconductor Manufacturing: Final Report on Findings from Benchmarking Eight-inch, Sub-350nm Wafer Fabrication Lines*. Publication of the Engineering Systems Research Center. 2002. URL http://books.google.de/books?id=TBLFtgAACAAJ.

Robert C. Leachman and D. Hodges. Benchmarking semiconductor manufacturing. *Semiconductor Manufacturing, IEEE Transactions on*, 9(2):158--169, 1996. ISSN 0894-6507. doi: 10.1109/66.492810. URL http://dx.doi.org/10.1109/66.492810.

Robert C. Leachman, Jeenyoung Kang, and Vincent Lin. Slim: Short cycle time and low inventory in manufacturing at samsung electronics. *Interfaces*, 32(1):61--77, 2002. doi: 10.1287/inte.32.1.61.15. URL http://dx.doi.org/10.1287/inte.32.1.61.15.

H. Todd LeBaron and Ruth Ann Hendrickson. Using emulation to validate a cluster tool simulation model. In J. A. Joines, Russell R. Barton, K. Kang, and Paul A. Fishwick, editors, *Proceedings of the Winter Simulation Conference*, WSC '00, 2000.

H. Todd LeBaron and Mark Pool. The simulation of cluster tools: a new semiconductor manufacturing technology. In J. D. Tew, M.S Manivannan, D. A. Sadowski, and Andrew F. Seila, editors, *Proceedings of the 26th Winter Simulation Conference*, pages 907--912, San Diego and CA and USA, 1994. Society for Computer Simulation International. ISBN 0-7803-2109-X.

Chung-Yee Lee and Reha Uzsoy. Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research*, 37(1):219--236, 1999. doi: 10.1080/002075499192020. URL http://dx.doi.org/10.1080/002075499192020.

Chung-Yee Lee, Reha Uzsoy, and Louis A. Martin-Vega. Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40(4):764--775, 1992. doi: 10.1287/opre.40.4.764. URL http://dx.doi.org/10.1287/opre.40.4.764.

Jon Lee. *First course in combinatorial optimization*. Cambridge University Press, New York, 2004. ISBN 978-0-511-18690-5.

Young-Hoon Lee and Michael L. Pinedo. Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100(3):464--474, 1997. ISSN 0377-2217. doi: 10.1016/S0377-2217(95)00376-2. URL http://dx.doi.org/10.1016/S0377-2217(95)00376-2.

J. K. Lenstra, A. H. G. Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. In Peter L. Hammer, E.L Johnson, B. H. Korte, and G. L. Nemhauser, editors, *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 343--362. Elsevier Science, 1977. doi: 10.1016/S0167-5060(08)70743-X. URL http://dx.doi.org/10.1016/S0167-5060(08)70743-X.

Joseph Y-T. Leung, editor. *Handbook of scheduling: Algorithms, models, and performance analysis*. Chapman & Hall/CRC, Boca Raton, 2004. ISBN 9780203489802.

Chung-Lun Li and Chung-Yee Lee. Scheduling with agreeable release times and due dates on a batch processing machine. *European Journal of Operational Research*, 96(3):564--569, 1997. ISSN 0377-2217. doi: 10.1016/0377-2217(95)00332-0. URL http://www.sciencedirect.com/science/article/pii/0377221795003320.

L. Li, F. Qiao, and Q. D. Wu. Aco-based scheduling of parallel batch processing machines to minimize the total weighted tardiness. In *Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on*, pages 280--285, 2009a. doi: 10.1109/COASE.2009.5234191. URL http://dx.doi.org/10.1109/COASE.2009.5234191.

Li Li and Fei Qiao. Aco-based scheduling for a single batch processing machine in semiconductor manufacturing. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 85--90, 2008. doi: 10.1109/COASE.2008.4626455. URL http://dx.doi.org/10.1109/COASE.2008.4626455.

Li Li, Fei Qiao, and Qidi Wu. Aco-based scheduling of parallel batch processing machines with incompatible job families to minimize total weighted tardiness. In Marco Dorigo, Mauro Birattari, Christian Blum, Maurice Clerc, Thomas Stützle, and Alan Winfield, editors, *Ant Colony Optimization and Swarm Intelligence*, volume 5217 of *Lecture Notes in Computer Science*, pages 219--226. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-87526-0. URL http://dx.doi.org/10.1007/978-3-540-87527-7_20.

Li Li, F. Qiao, and Q. D. Wu. Aco-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints. *The International Journal of Advanced Manufacturing Technology*, 44(9-10):985--994, 2009b. ISSN 0268-3768. doi: 10.1007/s00170-008-1904-8. URL http://dx.doi.org/10.1007/s00170-008-1904-8.

Shuguang Li. Makespan minimization on parallel batch processing machines with release times and job sizes. *Journal of Software*, 7(6), 2012. URL http://ojs.academypublisher.com/index.php/jsw/article/view/jsw070612031210.

Shuguang Li, Guojun Li, and Shaoqiang Zhang. Minimizing maximum lateness on identical parallel batch processing machines. In Kyung-Yong Chwa and J.IanJ Munro, editors, *Computing and Combinatorics*, volume 3106 of *Lecture Notes in Computer Science*, pages 229--237. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22856-1. doi: 10.1007/978-3-540-27798-9_26. URL http://dx.doi.org/10.1007/978-3-540-27798-9_26.

Shuguang Li, Guojun Li, Xiaoli Wang, and Qiming Liu. Minimizing makespan on a single batching machine with release times and non-identical job sizes. *Operations Research Letters*, 33(2):157--164, 2005a. ISSN 0167-6377. doi: 10.1016/j.orl.2004.04.009. URL http://dx.doi.org/10.1016/j.orl.2004.04.009.

Shuguang Li, Guojun Li, and Shaoqiang Zhang. Minimizing makespan with release times on identical parallel batching machines. *Discrete Applied Mathematics*, 148(1):127--134, 2005b. ISSN 0166-218X. doi: 10.1016/j.dam.2004.11.004. URL http://www.sciencedirect.com/science/article/pii/S0166218X04003695.

Wenhua Li, Zhenkun Zhang, and Sufang Yang. Online algorithms for scheduling unit length jobs on parallel-batch machines with lookahead. *Information Processing Letters*, 112(7):292--297, 2012a. ISSN 0020-0190. doi: 10.1016/j.ipl.2012.01.002. URL http://www.sciencedirect.com/science/article/pii/S0020019012000191.

Wenjie Li, Zhenkun Zhang, Hailing Liu, and Jinjiang Yuan. Online scheduling of equal-length jobs with incompatible families on multiple batch machines to maximize the weighted number of early jobs. *Information Processing Letters*, 112(12):503--508, 2012b. ISSN 0020-0190. doi: 10.1016/j.ipl.2012.03.015. URL http://dx.doi.org/10.1016/j.ipl.2012.03.015.

XiaoLin Li, YanLi Huang, Qi Tan, and Huaping Chen. Scheduling unrelated parallel batch processing machines with non-identical job sizes. *Computers & Operations Research*, 40(12):2983--2990, 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2013.06.016. URL http://www.sciencedirect.com/science/article/pii/S0305054813001731.

Da-Yin Liao and Chia-Nan Wang. Neural-network-based delivery time estimates for prioritized 300-mm automatic material handling operations. *IEEE Transactions on Semiconductor Manufacturing*, 17(3):324--332, 2004. ISSN 0894-6507. doi: 10.1109/TSM.2004.831533. URL http://dx.doi.org/10.1109/TSM.2004.831533.

Da-Yin Liao, Shi-Chung Chang, Kuo-Wei Pei, and Chi-Ming Chang. Daily scheduling for r&d semiconductor fabrication. *Semiconductor Manufacturing, IEEE Transactions on*, 9(4):550--561, 1996. ISSN 0894-6507. doi: 10.1109/66.542170. URL http://dx.doi.org/10.1109/66.542170.

Chee Peng Lim and Lakhmi C. Jain. Advances in swarm intelligence. In Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri, editors, *Innovations in swarm intelligence*. Springer, Berlin, 2009. ISBN 978-3-642-04224-9.

Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri, editors. *Innovations in swarm intelligence*. Springer, Berlin, 2009. ISBN 978-3-642-04224-9. doi: 10.1007/978-3-642-04225-6. URL http://dx.doi.org/10.1007/978-3-642-04225-6.

B. M. T. Lin and A. A. K. Jeng. Parallel-machine batch scheduling to minimize the maximum lateness and the number of tardy jobs. *International Journal of Production Economics*, 91(2):121--134, 2004. ISSN 0925-5273. doi: 10.1016/j.ijpe.2003.07.003. URL http://dx.doi.org/10.1016/j.ijpe.2003.07.003.

John D. C. Little. A proof for the queuing formula. *Operations Research*, 9(3):383--387, 1961. doi: 10.1287/opre.9.3.383. URL http://dx.doi.org/10.1287/opre.9.3.383.

L. L. Liu, C. T. Ng, and T. C. Edwin Cheng. Scheduling jobs with release dates on parallel batch processing machines. *Discrete Applied Mathematics*, 157(8):1825--1830, 2009. ISSN 0166-218X. doi: 10.1016/j.dam.2008.12.012. URL http://dx.doi.org/10.1016/j.dam.2008.12.012.

L. L. Liu, C. T. Ng, and T. C. Edwin Cheng. On scheduling unbounded batch processing machine(s). *Computers & Industrial Engineering*, 58(4):814--817, 2010. ISSN 0360-8352. doi: 10.1016/j.cie.2010.02.003. URL http://dx.doi.org/10.1016/j.cie.2010.02.003.

Lili Liu. *Batch scheduling problems*. PhD thesis, The Hong Kong Polytechnic University, Hong Kong, 2007. URL http://hdl.handle.net/10397/4138.

Lili Liu and Feng Zhang. Minimizing number of tardy jobs on a batch processing machine with incompatible job families. In *Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on*, volume 3, pages 277--280, 2008. doi: 10.1109/CCCM.2008.107. URL http://dx.doi.org/10.1109/CCCM.2008.107.

Zhaohui Liu and T. C. Edwin Cheng. Approximation schemes for minimizing total (weighted) completion time with release dates on a batch machine. *Theoretical Computer Science*, 347(1--2):288--298, 2005. ISSN 0304-3975. doi: 10.1016/j.tcs.2005.07.028. URL http://www.sciencedirect.com/science/article/pii/S0304397505004640.

Zhaohui Liu and Wenci Yu. Scheduling one batch processor subject to job release dates. *Discrete Applied Mathematics*, 105(1--3):129--136, 2000. ISSN 0166-218X. doi: 10.1016/S0166-218X(00)00181-5. URL http://www.sciencedirect.com/science/article/pii/S0166218X00001815.

Zhaohui Liu, Jinjiang Yuan, and T. C. Edwin Cheng. On scheduling an unbounded batch machine. *Operations Research Letters*, 31(1):42--48, 2003. ISSN 0167-6377. doi: 10.1016/S0167-6377(02)00186-4. URL http://www.sciencedirect.com/science/article/pii/S0167637702001864.

Helena R. Lourenco, Olivier C. Martin, and Thomas Stützle. Iterated local search: Framework and applications. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Di Lu, Hua-Ping Chen, Wen-Gong Zhang, and Rui Xu. An improved discrete particle swarm optimization algorithm for a single batch-processing machine with non-identical job sizes. In *Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD '09. 10th ACIS International Conference on*, pages 87--92, 2009a. doi: 10.1109/SNPD.2009.76. URL http://dx.doi.org/10.1109/SNPD.2009.76.

Lingfa Lu, T. C. Edwin Cheng, Jinjiang Yuan, and Liqi Zhang. Bounded single-machine parallel-batch scheduling with release dates and rejection. *Computers & Operations Research*, 36(10):2748--2751, 2009b. ISSN 0305-0548. doi: 10.1016/j.cor.2008.12.003. URL http://www.sciencedirect.com/science/article/pii/S0305054808002505.

Shenpeng Lu, Haodi Feng, and Xiuqian Li. Minimizing the makespan on a single parallel batching machine. *Theoretical Computer Science*, 411(7--9):1140--1145, 2010. ISSN 0304-3975. doi: 10.1016/j.tcs.2009.12.008. URL http://www.sciencedirect.com/science/article/pii/S0304397509008329.

Sean Luke. *Essentials of Metaheuristics*. 2009. URL http://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf.

M. Maier and E. Rechtin. *The Art of Systems Architecting*. Systems Engineering Series. CRC Press, 2000. ISBN 9780849304408. URL http://books.google.de/books?id=0T8NntOdjU0C.

Arnaud Malapert, Christelle Guéret, and Louis-Martin Rousseau. A constraint programming approach for a batch processing problem with non-identical job sizes. *European Journal of Operational Research*, 221(3):533--545, 2012. ISSN 0377-2217. doi: 10.1016/j.ejor.2012.04.008. URL http://www.sciencedirect.com/science/article/pii/S0377221712002846.

Sujay Malve and Reha Uzsoy. A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers & Operations Research*, 34(10):3016--3028, 2007. ISSN 0305-0548. doi: 10.1016/j.cor.2005.11.011. URL http://ac.els-cdn.com/S0305054805003588/1-s2.0-S0305054805003588-main.pdf?_tid=dc35e1c961fe2e3b1de8c79fe5c9910e&acdnat=1343142413_c91d5cfc59896ee3f3611dca61003698.

Rafael Marti and Gerhard Reinelt. *The linear ordering problem: Exact and heuristic methods in combinatorial optimization*. Springer, Heidelberg and and New York, 2011. ISBN 978-3-642-16729-4.

Rafael Marti, J. Marcos Moreno-Vega, and Abraham Duarte. Advanced multi-start methods. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Rafael Marti, Gerhard Reinelt, and Abraham Duarte. A benchmark library and a comparison of heuristic methods for the linear ordering problem. *Comput. Optim. Appl.*, 51(3):1297--1317, 2012. ISSN 0926-6003. doi: 10.1007/s10589-010-9384-9. URL http://dx.doi.org/10.1007/s10589-010-9384-9.

Rafael Marti, Mauricio G. C. Resende, and Celso C. Ribeiro. Multi-start methods for combinatorial optimization. *European Journal of Operational Research*, 226(1):1--8, 2013. ISSN 0377-2217. doi: 10.1016/j.ejor.2012.10.012. URL http://www.sciencedirect.com/science/article/pii/S0377221712007394.

D. P. Martin. Maximizing productivity improvements using short cycle time manufacturing (scm) concepts in a semiconductor manufacturing line. In *Advanced Semiconductor Manufacturing Conference and Workshop, 2000 IEEE/SEMI*, pages 63--67, 2000. doi: 10.1109/ASMC.2000.902559. URL http://dx.doi.org/10.1109/ASMC.2000.902559.

Donald P. Martin. The advantages of using short cycle time manufacturing (scm) instead of continuous flow manufacturing (cfm). In *Advanced Semiconductor Manufacturing Conference and Workshop, 1998. 1998 IEEE/SEMI*, pages 43--49, 1998. doi: 10.1109/ASMC.1998.731385. URL http://dx.doi.org/10.1109/ASMC.1998.731385.

Scott J. Mason, Mary E. Kurz, Michele E. Pfund, John W. Fowler, and L.M Pohl. Multi-objective semiconductor manufacturing scheduling: A random keys implementation of nsga-ii. In *Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling (SCIS '07)*, pages 159--164, 2007. doi: 10.1109/SCIS.2007.367684. URL http://dx.doi.org/10.1109/SCIS.2007.367684.

S. C. Mathewson. The application of program generator software and its extensions to discrete event simulation modeling. *IIE Transactions*, 16(1):3--18, 1984. ISSN 0740-817X. doi: 10.1080/07408178408974663. URL http://dx.doi.org/10.1080/07408178408974663.

M. Mathirajan and A. I. Sivakumar. Minimizing total weighted tardiness on heterogeneous batch processing machines with incompatible job families. *The International Journal of Advanced Manufacturing Technology*, 28(9-10):1038--1047, 2006a. ISSN 0268-3768. doi: 10.1007/s00170-004-2452-5. URL http://dx.doi.org/10.1007/s00170-004-2452-5.

M. Mathirajan and Appa Iyer Sivakumar. Scheduling of batch processors in semiconductor manufacturing -- a review. In *Innovation in Manufacturing Systems and Technology (IMST) 2003*. 2003. URL http://hdl.handle.net/1721.1/3755.

M. Mathirajan and Appa Iyer Sivakumar. A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *The International Journal of Advanced Manufacturing Technology*, 29(9):990--1001, 2006b. ISSN 0268-3768. URL http://dx.doi.org/10.1007/s00170-005-2585-1.

M. Mathirajan, A. I. Sivakumar, and P. Kalaivani. An efficient simulated annealing algorithm for scheduling burn-in oven with non-identical job-sizes. *The International Journal of Applied Management and Technology*, page 117, 2004.

Muthu Mathirajan, V. Bhargav, and V. Ramachandran. Minimizing total weighted tardiness on a batch-processing machine with non-agreeable release times and due dates. *The International Journal of Advanced Manufacturing Technology*, 48(9-12):1133--1148, 2010. ISSN 0268-3768. doi: 10.1007/s00170-009-2342-y. URL http://dx.doi.org/10.1007/s00170-009-2342-y.

Gary S. May and Costas J. Spanos. *Fundamentals of semiconductor manufacturing and process control*. Wiley-Interscience, Hoboken and NJ, 2006. ISBN 978-0-471-78406-7. URL http://dx.doi.org/10.1002/0471790281.

Kenneth N. McKay, Frank R. Safayeni, and John A. Buzacott. Job-shop scheduling theory: What is relevant? *Interfaces*, 18(4):84--90, 1988. URL http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=6690083&site=ehost-live.

Sanjay V. Mehta and Reha Uzsoy. Minimizing total tardiness on a batch processing machine with incompatible job families. *IIE Transactions*, 30(2):165--178, 1998. ISSN 0740-817X. doi: 10.1023/A:1007466101115. URL http://dx.doi.org/10.1023/A:1007466101115.

Sharif Melouk, Purushothaman Damodaran, and Ping-Yu Chang. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, 87(2):141--147, 2004. ISSN 0925-5273. doi: 10.1016/S0925-5273(03)00092-6. URL http://www.sciencedirect.com/science/article/pii/S0925527303000926.

Carlos A. Méndez, Jaime Cerdá, Ignacio E. Grossmann, Iiro Harjunkoski, and Marco Fahl. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering*, 30(6--7):913--946, 2006. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2006.02.008. URL http://www.sciencedirect.com/science/article/pii/S0098135406000287.

Jintao Meng and Xiaoxu Lu. The bounded single-machine parallel-batching scheduling problem with family jobs for minimizing makespan. *Procedia Environmental Sciences*, 10(0):374--378, 2011. ISSN 1878-0296. doi: 10.1016/j.proenv.2011.09.061. URL http://dx.doi.org/10.1016/j.proenv.2011.09.061.

Yuri Merkuryev. *Simulation-based case studies in logistics: Education and applied research*. Springer, London, 2009. ISBN 978-1-84882-186-6.

Manfred Mittler and Alexander K. Schoemig. Comparison of dispatching rules for semiconductor manufacturing using large facility models. In P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, editors, *Proceedings of the 31st Winter Simulation Conference*, WSC '99, pages 709--713, New York, 1999. ACM Press. ISBN 0-7803-5780-9. doi: 10.1145/324138.324463. URL http://dx.doi.org/10.1145/324138.324463.

Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097--1100, 1997. ISSN 0305-0548. doi: 10.1016/S0305-0548(97)00031-2. URL http://dx.doi.org/10.1016/S0305-0548(97)00031-2.

Vladimir Modrak and Pavol Semanco. Developments in modern operations management and cellular manufacturing. In Vladimir Modrak and R. Sudhakara Pandian, editors, *Operations management research and cellular manufacturing systems*, pages 1--20. Business Science Reference, Hershey PA, 2012. ISBN 978-1-61350-048-4.

Lars Mönch and Christian Almeder. Ant colony optimization approaches for scheduling jobs with incompatible families on parallel batch machines. In *Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2009)*. 2009.

Lars Mönch and Robert Unbehaun. Decomposition heuristics for minimizing earliness--tardiness on parallel burn-in ovens with a common due date. *Computers & Operations Research*, 34(11):3380--3396, 2007. ISSN 0305-0548. doi: 10.1016/j.cor.2006.02.003. URL http://dx.doi.org/10.1016/j.cor.2006.02.003.

Lars Mönch and Jens Zimmermann. Improving the performance of dispatching rules in semiconductor manufacturing by iterative simulation. In Ricki G. Ingalls, M. D. Rossetti, Jeffrey S. Smith, and B. A. Peters, editors, *Proceedings of the Winter Simulation Conference*, volume 2 of *WSC '04*, pages 1881--1887, 2004. doi: 10.1109/WSC.2004.1371544. URL http://dx.doi.org/10.1109/WSC.2004.1371544.

Lars Mönch, Oliver Rose, and Roland Sturm. A simulation framework for the performance assessment of shop-floor control systems. *SIMULATION*, 79(3):163--170, 2003. doi: 10.1177/0037549703256039. URL http://dx.doi.org/10.1177/0037549703256039.

Lars Mönch, Hari Balasubramanian, John W. Fowler, and Michele E. Pfund. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers & Operations Research*, 32(11):2731--2750, 2005. ISSN 0305-0548. doi: 10.1016/j.cor.2004.04.001. URL http://dx.doi.org/10.1016/j.cor.2004.04.001.

Lars Mönch, Robert Unbehaun, and You In Choung. Minimizing earliness--tardiness on a single burn-in oven with a common due date and maximum allowable tardiness constraint. *OR Spectrum*, 28(2):177--198, 2006a. doi: 10.1007/s00291-005-0013-4. URL http://dx.doi.org/10.1007/s00291-005-0013-4.

Lars Mönch, Jens Zimmermann, and Peter Otto. Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. *Engineering Applications of Artificial Intelligence*, 19(3):235--245, 2006b. ISSN 0952-1976. doi: 10.1016/j.engappai.2005.10.001. URL http://www.sciencedirect.com/science/article/pii/S0952197605001259.

Lars Mönch, John W. Fowler, Stéphane Dauzère-Pérès, Scott J. Mason, and Oliver Rose. Scheduling semiconductor manufacturing operations: Problems, solution techniques, and future challenges. In *Proceedings of the 4th Multidisciplinary International Scheduling Conference (MISTA)*, pages 192--201, 2009. URL http://www.mistaconference.org/2009/papers/192-201-104-P.pdf.

Lars Mönch, John W. Fowler, Stéphane Dauzère-Pérès, Scott J. Mason, and Oliver Rose. A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, 14(6):583--599, 2011a. ISSN 1094-6136. URL http://dx.doi.org/10.1007/s10951-010-0222-9.

Lars Mönch, Peter Lendermann, Leon F. McGinnis, and Arnd Schirrmann. A survey of challenges in modelling and decision-making for discrete event logistics systems. *Computers in Industry*, 62(6):557--567, 2011b. ISSN 0166-3615. doi: 10.1016/j.compind.2011.05.001. URL http://dx.doi.org/10.1016/j.compind.2011.05.001.

M. A. S. Monfared and J. B. Yang. Design of integrated manufacturing planning, scheduling and control systems: a new framework for automation. *The International Journal of Advanced Manufacturing Technology*, 33(5-6):545--559, 2007. ISSN 0268-3768. doi: 10.1007/s00170-006-0476-8. URL http://dx.doi.org/10.1007/s00170-006-0476-8.

Jairo R. Montoya-Torres. Manufacturing performance evaluation in wafer semiconductor factories. *The international journal of productivity and performance management : IJPPM.*, (55):300--310, 2006. doi: 10.1108/17410400610653246. URL http://dx.doi.org/10.1108/17410400610653246.

G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86 (1):82--85, 1998. ISSN 0018-9219. doi: 10.1109/JPROC.1998.658762. URL http://dx.doi.org/10.1109/JPROC.1998.658762.

James R. Morrison. Multiclass flow line models of semiconductor manufacturing equipment for fab-level simulation. *IEEE Transactions on Automation Science and Engineering*, 8(1):81--94, 2011a. doi: 10.1109/TASE.2010.2043733. URL http://dx.doi.org/10.1109/TASE.2010.2043733.

James R. Morrison. On the fidelity of the ax+b equipment model for clustered photolithography scanners in fab-level simulation. In S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and Michael C. Fu, editors, *Proceedings of the Winter Simulation Conference*, pages 2029--2039, 2011b. doi: 10.1109/WSC.2011.6147916. URL http://dx.doi.org/10.1109/WSC.2011.6147916.

Pablo Moscato and Carlos Cotta. A modern introduction to memetic algorithms. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Marcin Mosinski, Daniel Noack, F. S. Pappert, Oliver Rose, and Wolfgang Scholl. Cluster based analytical method for the lot delivery forecast in semiconductor fab with wide product range. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 1829--1839, 2011. doi: 10.1109/WSC.2011.6147897. URL http://dx.doi.org/10.1109/WSC.2011.6147897.

Mehrdad M. Moslehi, Richard A. Chapman, Man Wong, Ajit Paranjpe, Habib N. Najm, John Kuehne, Richard L. Yeakley, and Cecil J. Davis. Single-wafer integrated semiconductor device

processing. *IEEE Transactions on Electron Devices*, 39(1):4--32, 1992. doi: 10.1109/16.108208. URL http://dx.doi.org/10.1109/16.108208.

Ralph Mueller, Christos Alexopoulos, and Leon F. McGinnis. Automatic generation of simulation models for semiconductor manufacturing. In Shane G. Henderson, Bahar Biller, Ming-Hua Hsieh, John Shortle, D Tew Jeffrey, and Russell R. Barton, editors, *Proceedings of the 39th Winter Simulation Conference*, WSC '07, pages 648--657, Piscataway and NJ and USA, 2007. IEEE Press. ISBN 1-4244-1306-0.

Stephen Murray, John Geraghty, Paul Young, and Steve Sievwright. Time-limited next arrival heuristic for batch processing and setup reduction in a re-entrant environment. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2109--2117. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

Dima Nazzal and Douglas A. Bodner. A simulation-based design framework for automated material handling systems in 300mm fabrication facilities. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, WSC '03, pages 1351--1359. ACM Press, 2003. ISBN 0-7803-8132-7. URL http://dl.acm.org/citation.cfm?id=1030818.1030999.

Dima Nazzal and Ahmed El-Nashar. Survey of research in modeling conveyor-based automated material handling systems in wafer fabs. In Shane G. Henderson, Bahar Biller, Ming-Hua Hsieh, John Shortle, D Tew Jeffrey, and Russell R. Barton, editors, *Proceedings of the 39th Winter Simulation Conference*, WSC '07, pages 1781--1788, Piscataway and NJ and USA, 2007. IEEE Press. ISBN 1-4244-1306-0. URL http://dl.acm.org/citation.cfm?id=1351542.1351862.

John J. Neale and Izak Duenyas. Control of manufacturing networks which contain a batch processing machine. *IIE Transactions*, 32(11):1027--1041, 2000. ISSN 0740-817X. doi: 10.1023/A:1013780307222. URL http://dx.doi.org/10.1023/A%3A1013780307222.

R. F. Tavares Neto and M. Godinho Filho. Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*, 2012. ISSN 0952-1976. doi: 10.1016/j.engappai.2012.03.011. URL http://dx.doi.org/10.1016/j.engappai.2012.03.011.

Frank Neumann, Dirk Sudholt, and Carsten Witt. Computational complexity of ant colony optimization and its hybridization with local search. In Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri, editors, *Innovations in swarm intelligence*. Springer, Berlin, 2009. ISBN 978-3-642-04224-9.

Marcel F. Neuts. A general class of bulk queues with poisson input. *The Annals of Mathematical Statistics*, 38(3):p 759--770, 1967. URL http://www.jstor.org/stable/2238992.

Heiko Niedermayer and Oliver Rose. A simulation-based analysis of the cycle time of cluster tools. In Alexander Verbraeck and Vlatka Hlupic, editors, *Proceedings of 15th European Simulation Symposium*, pages 26--29, 2003.

Alexander G. Nikolaev and Sheldon H. Jacobson. Simulated annealing. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Daniel Noack. *Online Simulation in Semiconductor Manufacturing*. PhD thesis, Universität der Bundeswehr München, München, 2012.

Daniel Noack, Robert Kohn, Marcin Mosinski, Zhugen Zhou, Oliver Rose, Wolfgang Scholl, Peter Lendermann, and Boon-Ping Gan. Data modeling for online simulation - requirements and architecture. In *Proceedings of the 2010 FAIM Conference*, pages 1037--1044. 2010.

Daniel Noack, Marcin Mosinski, Oliver Rose, Peter Lendermann, Boon-Ping Gan, and Wolfgang Scholl. Challenges and solution approaches for the online simulation of semiconductor wafer fabs. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 1840--1851, 2011. doi: 10.1109/WSC.2011.6147898. URL http://dx.doi.org/10.1109/WSC.2011.6147898.

Q. Q. Nong, C. T. Ng, and T. C. Edwin Cheng. The bounded single-machine parallel-batching scheduling problem with family jobs and release dates to minimize makespan. *Operations Research Letters*, 36(1):61--66, 2008a. ISSN 0167-6377. doi: 10.1016/j.orl.2007.01.007. URL http://dx.doi.org/10.1016/j.orl.2007.01.007.

Qingqin Nong, Jinjiang Yuan, Ruyan Fu, Lin Lin, and Ji Tian. The single-machine parallel-batching on-line scheduling problem with family jobs to minimize makespan. *International Journal of Production Economics*, 111(2):435--440, 2008b. ISSN 0925-5273. doi: 10.1016/j.ijpe.2006.12.061. URL http://dx.doi.org/10.1016/j.ijpe.2006.12.061.

Peter Nyhuis and Hans-Peter Wiendahl. *Fundamentals of production logistics: Theory, tools and applications*. Springer, Berlin [u.a.], 2009. ISBN 978-3-540-34211-3.

Shigeru Obayashi, editor. *Evolutionary multi-criterion optimization: 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007 ; proceedings*. Springer, Berlin [u.a.], 2007. ISBN 978-3-540-70927-5.

Simon Oechsner and Oliver Rose. Scheduling cluster tools using filtered beam search and recipe comparison. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the 37th Winter Simulation Conference*, WSC '05, pages 2203--2210. ACM Press, 2005. ISBN 0-7803-9519-0.

Irfan M. Ovacik and Reha Uzsoy. Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup times. *International Journal of Production Research*, 33(11):3173--3192, 1995. doi: 10.1080/00207549508904867. URL http://dx.doi.org/10.1080/00207549508904867.

Onur Ozturk, Marie-Laure Espinouse, Maria Di Mascolo, and Alexia Gouin. Makespan minimisation on parallel batch processing machines with non-identical job sizes and release dates. *International Journal of Production Research*, 50(20):6022--6035, 2012. doi: 10.1080/00207543.2011.641358. URL http://dx.doi.org/10.1080/00207543.2011.641358.

José Antonio Parejo, A. Ruiz-Cortés, Sebastián Lozano, and Pablo Fernández. Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Computing*, 16(3):527--561, 2011. ISSN 1433-7479. doi: 10.1007/s00500-011-0754-8. URL http://dx.doi.org/10.1007/s00500-011-0754-8.

K. Park and James R. Morrison. Cluster tool design comparisons via simulation. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 1872--1882, 2011. doi: 10.1109/WSC.2011.6147901. URL http://dx.doi.org/10.1109/WSC.2011.6147901.

N. Rafiee Parsa, Behrooz Karimi, and Ali Husseinzadeh Kashan. A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes. *Computers & Operations Research*, 37(10):1720--1730, 2010. ISSN 0305-0548. doi: 10.1016/j.cor.2009.12.007. URL http://dx.doi.org/10.1016/j.cor.2009.12.007.

Vangelis Th. Paschos. *Combinatorial optimization and theoretical computer science: Interfaces and perspectives : 30th anniversary of the LAMSADE*. ISTE and Wiley, London and Hoboken and NJ, 2008. ISBN 978-1-84821-021-9.

Ray J. Paul and David W. Balmer. *Simulation modelling*. Chartwell-Bratt, Bromley, 1993. ISBN 0862382807.

Mohammad Mahdi Paydar and Mohammad Saidi-Mehrabad. A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. *Computers & Operations Research*, 40(4):980--990, 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2012.10.016. URL http://www.sciencedirect.com/science/article/pii/S0305054812002286.

Jula Payman and Robert C. Leachman. Coordinated multistage scheduling of parallel batch-processing machines under multiresource constraints. *Operations Research*, 58(4-Part-1):933--947, 2010. doi: 10.1287/opre.1090.0788. URL http://dx.doi.org/10.1287/opre.1090.0788.

Giulia Pedrielli, Tolio Tullio, Walter Terkaj, and Marco Sacco. Distributed modeling of discrete event systems. In Eldin Wee Chuan Lim, editor, *Discrete Event Simulations*, pages 3--46. InTech, 2012. ISBN 978-953-51-0741-5. doi: 10.5772/50350. URL http://dx.doi.org/10.5772/50350.

Imelda C. Perez, John W. Fowler, and W. Matthew Carlyle. Minimizing total weighted tardiness on a single batch process machine with incompatible job families. *Computers & Operations Research*, 32(2):327--341, 2005. ISSN 0305-0548. doi: 10.1016/S0305-0548(03)00239-9. URL http://dx.doi.org/10.1016/S0305-0548(03)00239-9.

Terry L. Perkinson, Peter K. McLarty, Ronald S. Gyurcsik, and Ralph K. Cavin. Single-wafer cluster tool performance: an analysis of throughput. *IEEE Transactions on Semiconductor Manufacturing*, 7(3):369--373, 1994. ISSN 0894-6507. doi: 10.1109/66.311340. URL http://dx.doi.org/10.1109/66.311340.

Terry L. Perkinson, Ronald S. Gyurcsik, and Peter K. McLarty. Single-wafer cluster tool performance: an analysis of the effects of redundant chambers and revisitation sequences on throughput. *IEEE Transactions on Semiconductor Manufacturing*, 9(3):384--400, 1996. ISSN 0894-6507. doi: 10.1109/66.536110. URL http://dx.doi.org/10.1109/66.536110.

Anna Persson, Henrik Grimm, Amos Ng, Thomas Lezama, Jonas Ekberg, Stephan Falk, and Peter Stablum. Simulation-based multi-objective optimization of a real-world scheduling problem. In *Proceedings of the 38th Conference on Winter Simulation*, WSC '06, pages 1757--1764. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7. URL http://dl.acm.org/citation.cfm?id=1218112.1218431.

Mikel D. Petty. Verification and validation. In John A. Sokolowski and Catherine M. Banks, editors, *Principles of modeling and simulation*, pages 121--147. John Wiley, Hoboken and N.J, 2009. ISBN 978-0-470-28943-3.

Nipa Phojanamongkolkij and Omar Ghrayeb. Batch size determination for wafer fabrication using genetic algorithms. *Revista Ingeniería Industrial*, 4(1):5--12, 2005. ISSN 07179103. URL http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=23291275&site=ehost-live.

Nipa Phojanamongkolkij, John W. Fowler, and Jeffery K. Cochran. Determining operating criterion of batch processing operations for wafer fabrication. *Journal of Manufacturing Systems*, 21 (5):363--379, 2002. ISSN 0278-6125. doi: 10.1016/S0278-6125(02)80035-8. URL http://www.sciencedirect.com/science/article/pii/S0278612502800358.

Neal G. Pierce and Michael J. Drevna. Development of generic simulation models to evaluate wafer fabrication cluster tools. In J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, editors, *Proceedings of the 24th Winter Simulation Conference*, WSC '92, pages 874--878. ACM Press, 1992.

Devadas Pillai. The future of semiconductor manufacturing. *IEEE Robotics Automation Magazine*, 13(4):16--24, 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.250560. URL http://dx.doi.org/10.1109/MRA.2006.250560.

Michael L. Pinedo. Scheduling, stochastic scheduling, and online deterministic scheduling: A comparative overview. In Joseph Y-T. Leung, editor, *Handbook of scheduling*. Chapman & Hall/CRC, Boca Raton, 2004. ISBN 9780203489802.

Michael L. Pinedo. *Planning and scheduling in manufacturing and services*. Springer, New York and NY, 2005. ISBN 0-387-22198-0.

Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Science+Business Media LLC, New York and NY, third edition edition, 2008. ISBN 9780387789347.

David Pisinger and Stefan Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Pisut Pongchairrerks and Voratas Kachitvichyanukul. A comparison between algorithms vns with pso and vns without pso for job-shop scheduling problems. *International Journal of Computational Science*, 1(2):179--191, 2007.

Chung Poon and Pixing Zhang. Minimizing makespan in batch machine scheduling. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, D. Lee, and Shang-Hua Teng, editors, *Algorithms and Computation*, volume 1969 of *Lecture Notes in Computer Science*, pages 189--234. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-41255-7. URL http://dx.doi.org/10.1007/3-540-40996-3_33.

Chung Keung Poon and Wenci Yu. On minimizing total completion time in batch machine scheduling. *International Journal of Foundations of Computer Science*, 15(04):593--607, 2004. doi: 10.1142/S0129054104002637. URL http://dx.doi.org/10.1142/S0129054104002637.

V. William Porto. Evolutionary programming. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

J. Potoradi, Ong Siong Boon, and Scott J. Mason. Using simulation-based scheduling to maximize demand fulfillment in a semiconductor assembly facility. In *Simulation Conference, 2002. Proceedings of the Winter*, volume 2, pages 1857--1861 vol.2, 2002. doi: 10.1109/WSC.2002.1166479. URL http://dx.doi.org/10.1109/WSC.2002.1166479.

Juergen Potoradi and Gerald Winz. Determining optimal lot-size for a semiconductor back-end factory. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 720--726 vol.1, 1999. doi: 10.1109/WSC.1999.823202. URL http://dx.doi.org/10.1109/WSC.1999.823202.

Chris N. Potts and Mikhail Y. Kovalyov. Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):228--249, 2000. ISSN 0377-2217. doi: 10.1016/S0377-2217(99)00153-8. URL http://dx.doi.org/10.1016/S0377-2217(99)00153-8.

Chris N. Potts and V. A. Strusevich. Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society*, 60(S1):41--68, 2009. doi: 10.1057/jors.2009.2. URL http://dx.doi.org/10.1057/jors.2009.2.

Chris N. Potts and L. N. van Wassenhove. Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. *J Oper Res Soc*, 43(5):395--406, 1992. ISSN 0160-5682. URL http://dx.doi.org/10.1057/jors.1992.66.

Steven Prestwich. The relation between complete and incomplete search. In Christian Blum, Maria José Blesa Aguilera, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, Studies in computational intelligence. Springer, Berlin and Heidelberg, 2008. ISBN 9783540782940.

Kirk Pruhs, Jirı Sgall, and Eric Torng. Online scheduling. In Joseph Y-T. Leung, editor, *Handbook of scheduling*, pages 335--377. Chapman & Hall/CRC, Boca Raton, 2004. ISBN 9780203489802.

Jakob Puchinger and Günther R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In *Proceedings of the First international work-conference on the Interplay Between Natural and Artificial Computation conference on Artificial Intelligence and Knowledge Engineering Applications: a bioinspired approach - Volume Part II*, IWINAC'05, pages 41--53, Berlin and Heidelberg, 2005. Springer-Verlag US. ISBN 3-540-26319-5. doi: 10.1007/11499305_5. URL http://dx.doi.org/10.1007/11499305_5.

Li Li Qiao, Fei Ma Yumin Qiao, and Kai Ye. Simulation-based modular scheduling system of semiconductor manufacturing. 2012a. URL http://www.intechopen.com/books/export/citation/BibTex/production-scheduling/simulation-based-modular-scheduling-system-of-semiconductor-manufacturing.

Yan Qiao, NaiQi Wu, and MengChu Zhou. Petri net modeling and wafer sojourn time analysis of single-arm cluster tools with residency time constraints and activity time variation. *Semiconductor Manufacturing, IEEE Transactions on*, 25(3):432--446, 2012b. ISSN 0894-6507. doi: 10.1109/TSM.2012.2199338. URL http://dx.doi.org/10.1109/TSM.2012.2199338.

T. Quinn and E. Bass. 300 mm factory layout and material handling modeling: Phase i report: Tech transfer document # 99023688b-eng, 1999.

N. R. Srinivasa Raghavan and M. Venkataramana. Scheduling parallel batch processors with incompatible job families using ant colony optimization. In *Proceedings of the International Conference on Automation Science and Engineering*, pages 507--512, Shanghai, 2006. doi: 10.1109/COASE.2006.326933. URL http://dx.doi.org/10.1109/COASE.2006.326933.

Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334--350, 2001. ISSN 1066-8888. doi: 10.1007/s007780100057. URL http://dx.doi.org/10.1007/s007780100057.

Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23:2000, 2000.

Günther R. Raidl, Jakob Puchinger, and Christian Blum. Metaheuristic hybrids. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Lars G. Randell and Gunnar S. Bolmsjö. Database driven factory simulation: a proof-of-concept demonstrator. In B. A. Peters, Jeffrey S. Smith, D. J. Medeiros, and M. W. Rohrer, editors, *Proceedings of the 33nd Winter Simulation Conference (WSC)*, WSC '01, pages 977--983, Washington and DC and USA, 2001. IEEE Computer Society Press. ISBN 0-7803-7309-X. URL http://dl.acm.org/citation.cfm?id=564124.564263.

A. Ravi Ravindran, editor. *Operations research and management science handbook*. CRC Press, Boca Raton, 2008. ISBN 978-0849397219.

Colin R. Reeves. Genetic algorithms. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Dirk Reichelt and Lars Mönch. Multiobjective scheduling of jobs with incompatible families on parallel batch machines. In Jens Gottlieb and Günther R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 209--221. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33178-0. doi: 10.1007/11730095_18. URL http://dx.doi.org/10.1007/11730095_18.

H. A. Reijers and W. M. P. van der Aalst. Short-term simulation: Bridging the gap between operational control and strategic decision making. In *Proceedings of the IASTED International Conference*. 1999.

Edward M. Reingold. Algorithm design and analysis techniques. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and theory of computation handbook*. CRC Press, Boca Raton [u.a.], 2010. ISBN 978-1-58488-822-2.

Mauricio G. C. Resende and Celso C. Ribeiro. Greedy randomized adaptive search procedures. *Handbook of metaheuristics*, pages 219--249, 2003.

Mauricio G. C. Resende and Celso C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Mauricio G. C. Resende, Celso C. Ribeiro, Fred Glover, and Rafael Marti. Scatter search and path-relinking: Fundamentals, advances, and applications. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Margarita Reyes-Sierra and Carlos A. Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *INTERNATIONAL JOURNAL OF COMPUTATIONAL INTELLIGENCE RESEARCH*, 2(3):287--308, 2006.

Julian Richards. Impacts of imminent changes in the semiconductor industry, 2013.

N. Robertson and T. Perera. Automated data collection for simulation? *Simulation Practice and Theory*, 9(6--8):349--364, 2002. ISSN 0928-4869. doi: 10.1016/S0928-4869(01)00055-6. URL http://dx.doi.org/10.1016/S0928-4869(01)00055-6.

J. Robinson, John W. Fowler, and Jonathan F. Bard. The use of upstream and downstream information in scheduling semiconductor batch operations. *International Journal of Production Research*, 33(7):1849--1869, 1995. doi: 10.1080/00207549508904785. URL http://dx.doi.org/10.1080/00207549508904785.

J. Robinson, John W. Fowler, and Jonathan F. Bard. A review of real-time control strategies for furnace batch sizing in semiconductor manufacturing. *Jennifer K. Robinsons Consulting*, 2000.

S. Robinson and V. Bhatia. Secrets of successful simulation projects. In *Simulation Conference Proceedings, 1995. Winter*, pages 61--67, 1995. doi: 10.1109/WSC.1995.478706. URL http://dx.doi.org/10.1109/WSC.1995.478706.

Oliver Rose. Wip evolution of a semiconductor factory after a bottleneck workcenter breakdown. In D. J. Medeiros, E.F Watson, J.S Carson, and M.S Manivannan, editors, *Proceedings of the 30th Winter Simulation Conference*, WSC '98, pages 997--1004, Los Alamitos and CA and USA, 1998. IEEE Computer Society Press. ISBN 0-7803-5134-7. URL http://dx.doi.org/10.1109/WSC.1998.745805.

Oliver Rose. Conload - a new lot release rule for semiconductor wafer fabs. In P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, editors, *Proceedings of the 31st Winter Simulation Conference*, WSC '99, pages 850--855, New York, 1999a. ACM Press. ISBN 0-7803-5780-9. doi: 10.1145/324138.324535. URL http://dx.doi.org/10.1145/324138.324535.

Oliver Rose. Estimation of the cycle time distribution of a wafer fab by a simple simulation model. In *SMOMS '99 (1999 WMC)*, pages 133--138, San Francisco and CA, 1999b. URL http://www3.informatik.uni-wuerzburg.de/staff/rose/papers/smoms99.pdf.

Oliver Rose. Why do simple wafer fab models fail in certain scenarios? In J. A. Joines, Russell R. Barton, K. Kang, and Paul A. Fishwick, editors, *Proceedings of the Winter Simulation Conference*, volume 2 of *WSC '00*, pages 1481 --1490 vol.2, 2000. doi: 10.1109/WSC.2000.899129. URL http://dx.doi.org/10.1109/WSC.2000.899129.

Oliver Rose. The shortest processing time first (sptf) dispatch rule and some variants in semiconductor manufacturing. In B. A. Peters, Jeffrey S. Smith, D. J. Medeiros, and M. W. Rohrer, editors, *Proceedings of the 33nd Winter Simulation Conference (WSC)*, WSC '01, pages 1220--1224, Washington and DC and USA, 2001. IEEE Computer Society Press. ISBN 0-7803-7309-X.

Oliver Rose. Some issues of the critical ratio dispatch rule in semiconductor manufacturing. In Enver Yücesan, C. H. Chen, Jane L. Snowdon, and J. M. Charnes, editors, *Proceedings of the 34th Winter simulation Conference*, WSC '02, pages 1401--1405. Winter Simulation Conference, 2002. ISBN 0-7803-7615-3.

Oliver Rose. Accelerating products under due-date oriented dispatching rules in semiconductor manufacturing. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, volume 2 of *WSC '03*, pages 1346--1350. ACM Press, 2003a. ISBN 0-7803-8132-7. doi: 10.1109/WSC.2003.1261571. URL http://dx.doi.org/10.1109/WSC.2003.1261571.

Oliver Rose. Comparison of due-date oriented dispatch rules in semiconductor manufacturing. In *Proceedings of the Industrial Engineering Research Conference*, pages 18--20, 2003b.

Oliver Rose. Modeling tool failures in semiconductor fab simulation. In Ricki G. Ingalls, M. D. Rossetti, Jeffrey S. Smith, and B. A. Peters, editors, *Proceedings of the Winter Simulation Conference*, WSC '04, pages 1910--1914, 2004.

Oliver Rose. Implementation of a simulation-based optimizer for semiconductor wafer factories. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA '06)*, pages 943--949, 2006. doi: 10.1109/ETFA.2006.355198. URL http://dx.doi.org/10.1109/ETFA.2006.355198.

Oliver Rose. Improved simple simulation models for semiconductor wafer factories. In Shane G. Henderson, Bahar Biller, Ming-Hua Hsieh, John Shortle, D Tew Jeffrey, and Russell R. Barton, editors, *Proceedings of the 39th Winter Simulation Conference*, WSC '07, pages 1708--1712, Piscataway and NJ and USA, 2007. IEEE Press. ISBN 1-4244-1306-0.

V. Roshanaei, B. Naderi, Fariborz Jolai, and M. Khalili. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems*, 25(6):654--661, 2009. ISSN 0167-739X. doi: 10.1016/j.future.2009.01.004. URL http://dx.doi.org/10.1016/j.future.2009.01.004.

Guenter Rudolph. Evolution strategies. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Pub. and Oxford University Press, Bristol and and Philadelphia and New York, 1997. ISBN 0-7503-0392-1.

Jeffrey L. Rummel. An empirical investigation of costs in batching decisions*. *Decision Sciences*, 31(1):79--103, 2000. ISSN 1540-5915. doi: 10.1111/j.1540-5915.2000.tb00925.x. URL http://dx.doi.org/10.1111/j.1540-5915.2000.tb00925.x.

David Ruppert, Lee Schruben, Mike Freimer, and (Keine Angabe). Meta-modeling of a cluster tool simulator. In *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM)*. 2000.

Stuart J. Russell and Peter Norvig. *Artificial intelligence: A modern approach*. Prentice Hall, Englewood Cliffs and N.J, 1995. ISBN 0-13-103805-2.

R. W. Saaty. The analytic hierarchy process - what it is and how it is used. *Mathematical Modelling*, 9:161--176, 1987. ISSN 0270-0255. doi: 10.1016/0270-0255(87)90473-8. URL http://dx.doi.org/10.1016/0270-0255(87)90473-8.

Thomas L. Saaty. How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, 48(1):9--26, 1990. ISSN 0377-2217. doi: 10.1016/0377-2217(90)90057-I. URL http://www.sciencedirect.com/science/article/pii/037722179090057I.

Thomas L. Saaty. Highlights and critical points in the theory and application of the analytic hierarchy process. *European Journal of Operational Research*, 74(3):426--447, 1994. ISSN 0377-2217. doi: 10.1016/0377-2217(94)90222-4. URL http://www.sciencedirect.com/science/article/pii/0377221794902224.

M. T. Yazdani Sabouni and Fariborz Jolai. Optimal methods for batch processing problem with makespan and maximum lateness objectives. *Applied Mathematical Modelling*, 34(2):314--324, 2010. ISSN 0307-904X. doi: 10.1016/j.apm.2009.04.007. URL http://dx.doi.org/10.1016/j.apm.2009.04.007.

R. Sahraeian, F. Samaei, and I. Rastgar. Minimizing the makespan on parallel batch scheduling with stochastic times. In J. Carlos Prado-Prado and Jesús García-Arca, editors, *Annals of Industrial Engineering 2012*, pages 209--216. Springer London, 2014. ISBN 978-1-4471-5348-1. doi: 10.1007/978-1-4471-5349-8_25. URL http://dx.doi.org/10.1007/978-1-4471-5349-8_25.

Robert G. Sargent. Verification and validation of simulation models. In *Proceedings of the 2010 Winter Simulation Conference, WSC 2010, Baltimore, Maryland, USA, 5-8 December 2010*, pages 166--183. Winter Simulation Conference, 2010.

Robert R. Schaller. *Technological Innovation in the Semiconductor Industry: A Case Study of the International Technology Roadmap for Semiconductors (ITRS)*. George Mason University, 2004.

Roland Schelasin. Using static capacity modeling and queuing theory equations to predict factory cycle time performance in semiconductor manufacturing. In *1D- WSC2011*, pages 2040--2049, 2011. doi: 10.1109/WSC.2011.6147917. URL http://dx.doi.org/10.1109/WSC.2011.6147917.

Kilian Schmidt, Jörg Weigang, and Oliver Rose. Modeling semiconductor tools for small lotsize fab simulations. In L. Felipe Perrone, Barry Lawson, Jason Liu, and Frederick P. Wieland, editors, *Proceedings of the Winter Simulation Conference*, WSC '06, pages 1811--1816. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7. doi: 10.1109/WSC.2006.322959. URL http://dx.doi.org/10.1109/WSC.2006.322959.

Wolfgang Scholl. Coping with typical unpredictable incidents in a logic fab. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 2030--2034. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6. URL http://dl.acm.org/citation.cfm?id=1516744.1517098.

Wolfgang Scholl and J. Domaschke. Implementation of modeling and simulation in semiconductor wafer fabrication with time constraints between wet etch and furnace operations. In *Semiconductor Manufacturing Conference Proceedings, 1999 IEEE International Symposium on*, pages 61--63, 1999. doi: 10.1109/ISSM.1999.808738. URL http://dx.doi.org/10.1109/ISSM.1999.808738.

Wolfgang Scholl and J. Domaschke. Implementation of modeling and simulation in semiconductor wafer fabrication with time constraints between wet etch and furnace operations. *Semiconductor Manufacturing, IEEE Transactions on*, 13(3):273--277, 2000. ISSN 0894-6507. doi: 10.1109/66.857935. URL http://dx.doi.org/10.1109/66.857935.

Wolfgang Scholl, Boon-Ping Gan, Daniel Noack, Oliver Rose, Ming Li Peh, Peter Lendermann, and Patrick Preuss. Towards realization of a high-fidelity simulation model for short-term horizon forecasting in wafer fabrication facilities. In *Simulation Conference (WSC), Proceedings of the 2010 Winter, title=*, pages 2563--2574, 2010. doi: 10.1109/WSC.2010.5678952. URL http://dx.doi.org/10.1109/WSC.2010.5678952.

Wolfgang Scholl, Boon-Ping Gan, Peter Lendermann, Daniel Noack, Oliver Rose, P. Preuss, and F. S. Pappert. Implementation of a simulation-based short-term lot arrival forecast in a mature 200mm semiconductor fab. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 1927--1938, 2011. doi: 10.1109/WSC.2011.6147907. URL http://dx.doi.org/10.1109/WSC.2011.6147907.

SEMI. Amd fab30 - an overview of factory automation requirements and design: Aux 0002-0600, 2000.

SEMI E10. Specification for definition and easurement of equipment reliability, availability, and maintainability (ram). specification for definition and easurement of equipment reliability, availability, and maintainability (ram), 2000.

SEMI E101. Provisional guide for efem functional structure model. provisional guide for efem functional structure model, 2000.

SEMI E102. Provisional specification for cim framework material transport and storage component. provisional specification for cim framework material transport and storage component, 2000.

SEMI E105. Provisional specification for cim framework scheduling component. provisional specification for cim framework scheduling component, 2000.

SEMI E106. Provisional overview guide to semi standards for physical interfaces and carriers for 300 mm wafers. provisional overview guide to semi standards for physical interfaces and carriers for 300 mm wafers, 2000.

SEMI E11. Semiconductor equipment and materials international (semi). guideline for 125 mm, 150 mm , and 200 mm plastic and metal carrier application. guideline for 125 mm, 150 mm , and 200 mm plastic and metal carrier application, 2000.

SEMI E23. Specification for cassette transfer parallel i/o interface. specification for cassette transfer parallel i/o interface, 2000.

SEMI E30. Generic model for communications and control of manufacturing equipment (gem). generic model for communications and control of manufacturing equipment (gem), 2000.

SEMI E4. Semi equipment communications standard 1 message transfer (secs-i). semi equipment communications standard 1 message transfer (secs-i), 2000.

SEMI E47. Provisional mechanical specification for boxes and pods used to transport and store 300 mm wafers. provisional mechanical specification for boxes and pods used to transport and store 300 mm wafers, 2000.

SEMI E5. Semiconductor equipment and materials international (semi). semi equipment communications standard 2 message content (secs-ii). semi equipment communications standard 2 message content (secs-ii), 2000.

SEMI E81. Provisional specification for cim framework domain architecture, 2000.

SEMI E84. Specification for enhanced carrier handoff parallel i/o interface. specification for enhanced carrier handoff parallel i/o interface, 2000.

SEMI E88. Specification for amhs storage sem (stocker sem). specification for amhs storage sem (stocker sem), 2000.

SEMI E96. Guide for cim framework technical architecture: Guide for cim framework technical architecture, 2000.

SEMI International Standards. Compilation of terms: Semiconductor equipment and materials international, 2009.

Mehmet Sevkli and Mehmet E. Aydin. A variable neighbourhood search algorithm for job shop scheduling problems. In Jens Gottlieb and Günther R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 261--271. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33178-0. URL http://dx.doi.org/10.1007/11730095_22.

Mehmet Sevkli and Mehmet E. Aydin. Parallel variable neighbourhood search algorithms for job shop scheduling problems. *IMA Journal of Management Mathematics*, 18(2):117--133, 2007.

Mehmet Sevkli and H. Uysal. A modified variable neighborhood search for minimizing the makespan on identical parallel machines. In *Computers Industrial Engineering, 2009. CIE 2009. International Conference on*, pages 108--111, 2009. doi: 10.1109/ICCIE.2009.5223485. URL http://dx.doi.org/10.1109/ICCIE.2009.5223485.

D. Y. Sha, S. Y. Hsu, and X. D. Lai. Study of look-ahead batching rule. In *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference 2004*. 2004.

D. Y. Sha, S. Y. Hsu, Z. H. Che, and C. H. Chen. A dispatching rule for photolithography scheduling with an on-line rework strategy. *Computers & Industrial Engineering*, 50(3):233--247, 2006. ISSN 0360-8352. URL http://dx.doi.org/10.1016/j.cie.2006.04.002.

D. Y. Sha, Sheng-Yuan Hsu, and X. D. Lai. Design of due-date oriented look-ahead batching rule in wafer fabrication. *The International Journal of Advanced Manufacturing Technology*, 35(5-6): 596--609, 2007. ISSN 0268-3768. doi: 10.1007/s00170-006-0723-z. URL http://dx.doi.org/10.1007/s00170-006-0723-z.

J. George Shanthikumar, Shengwei Ding, and Mike Tao Zhang. Queueing theory for semiconductor manufacturing systems: A survey and open problems. *IEEE Transactions on Automation Science and Engineering*, 4(4):513--522, 2007. doi: 10.1109/TASE.2007.906348. URL http://dx.doi.org/10.1109/TASE.2007.906348.

Hao Shao, Hua-Ping Chen, G.Q Huang, Rui Xu, Ba-Yi Cheng, Shuan-shi Wang, and Bo-Wen Liu. Minimizing makespan for parallel batch processing machines with non-identical job sizes using neural nets approach. In *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on*, pages 1921--1924, 2008a. doi: 10.1109/ICIEA.2008.4582854. URL http://dx.doi.org/10.1109/ICIEA.2008.4582854.

Hao Shao, Hua-Ping Chen, Rui Xu, and Wen-Gong Zhang. Minimizing makespan for a single batch processing machine with non-identical job sizes using neural nets approach. In *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, volume 6, pages 511--514, 2008b. doi: 10.1109/ICNC.2008.368. URL http://dx.doi.org/10.1109/ICNC.2008.368.

Sameer T. Shikalgar, D. Fronckowiak, and E.A MacNair. Application of cluster tool modeling to a 300 mm fab simulation. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, volume 2 of *WSC '03*, pages 1394 -- 1397 vol.2. ACM Press, 2003. ISBN 0-7803-8132-7. doi: 10.1109/WSC.2003.1261581. URL http://dx.doi.org/10.1109/WSC.2003.1261581.

John Silberholz and Bruce Golden. Comparison of metaheuristics. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

R. Singh, M. Fakhruddin, and K.F Poole. The impact of single-wafer processing on semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 16(2):96--101, 2003. ISSN 0894-6507. doi: 10.1109/TSM.2003.810941. URL http://dx.doi.org/10.1109/TSM.2003.810941.

Michael Sipser. *Introduction to the theory of computation*. Course technology, Boston, 2 edition, 2006. ISBN 0-534-95097-3.

A. I. Sivakumar, Chao Qi, and A. Hidayat. Experimental study on variations of wipload control in semiconductor wafer fabrication environment. In *Simulation Conference, 2008. WSC 2008. Winter*, pages 2035--2040, 2008. doi: 10.1109/WSC.2008.4736299. URL http://dx.doi.org/10.1109/WSC.2008.4736299.

Appa Iyer Sivakumar. Optimization of a cycle time and utilization in semiconductor test manufacturing using simulation based, on-line, near-real-time scheduling system. In P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, editors, *Proceedings of the 31st Winter Simulation Conference*, WSC '99, pages 727--735, New York, 1999. ACM Press. ISBN 0-7803-5780-9. doi: 10.1145/324138.324467. URL http://dx.doi.org/10.1145/324138.324467.

Anders Skoogh and Björn Johansson. A methodology for input data management in discrete event simulation projects. In Scott J. Mason, Raymond R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler, editors, *Proceedings of the 40th Winter Simulation Conference*, WSC '08, pages 1727--1735. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

J. S. Smith, R. A. Wysk, D. T. Sturrock, S. E. Ramaswamy, G. D. Smith, and S. B. Joshi. Discrete event simulation for shop floor control. In *Simulation Conference Proceedings, 1994. Winter*, pages 962--969, 1994. doi: 10.1109/WSC.1994.717475. URL http://dx.doi.org/10.1109/WSC.1994.717475.

John A. Sokolowski. Simulation: Models that vary over time. In John A. Sokolowski and Catherine M. Banks, editors, *Principles of modeling and simulation*, pages 47--69. John Wiley, Hoboken and N.J, 2009. ISBN 978-0-470-28943-3.

John A. Sokolowski and Catherine M. Banks, editors. *Principles of modeling and simulation: A multidisciplinary approach*. John Wiley, Hoboken and N.J, 2009. ISBN 978-0-470-28943-3.

John A. Sokolowski and Catherine M. Banks. *Modeling and simulation in the medical and health sciences*. Wiley, Hoboken, 2011. ISBN 978-0-470-76947-8.

Lance Solomon, John W. Fowler, Michele E. Pfund, and Paul H. Jensen. The inclusion of future arrivals and downstream setups into wafer fabrication batch processing decisions. *Journal of Electronics Manufacturing*, 11(02):149--159, 2002. doi: 10.1142/S0960313102000370. URL http://dx.doi.org/10.1142/S0960313102000370.

Young Jun Son and Richard A. Wysk. Automatic simulation model generation for simulation-based, real-time shop floor control. *Computers in Industry*, 45(3):291--308, 2001. ISSN 0166-3615. doi: 10.1016/S0166-3615(01)00086-0. URL http://dx.doi.org/10.1016/S0166-3615(01)00086-0.

Craig Stevens and Tony Jakubiec. Wafer processing architecture including single-wafer load lock with cooling unit, 2002. URL http://www.freepatentsonline.com/6431807.html.

William J. Stewart. *Probability, Markov chains, queues, and simulation: The mathematical basis of performance modeling*. Princeton University Press, Princeton and N.J, 2009. ISBN 978-0-691-14062-9.

Kilian Stubbe. *Development and Simulation Assessment of Semiconductor Production System Enhancements for Fast Cycle Times*. PhD thesis, TECHNISCHEN UNIVERSITÄT DRESDEN, Dresden, 2010. URL http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-27343.

Yu-sheng Su, Chiang-jen Peng, Pin-chia Su, and Wen-lang Wu. Wafer transfer robot having wafer blades equipped with sensors, 2004. URL http://www.freepatentsonline.com/6808589.html.

Chang Sup Sung and Y. I. Choung. A neural network approach for batching decisions in wafer fabrication. *International Journal of Production Research*, 37(13):3101--3114, 1999. doi: 10.1080/002075499190437. URL http://dx.doi.org/10.1080/002075499190437.

Chang Sup Sung and Y. I. Choung. Minimizing makespan on a single burn-in oven in semiconductor manufacturing. *European Journal of Operational Research*, 120(3):559--574, 2000. ISSN 0377-2217. doi: 10.1016/S0377-2217(98)00391-9. URL http://www.sciencedirect.com/science/article/pii/S0377221798003919.

Chang Sup Sung and J. I. Min. Scheduling in a two-machine flowshop with batch processing machine(s) for earliness/tardiness measure under a common due date. *European Journal of Operational Research*, 131(1):95--106, 2001. ISSN 0377-2217. doi: 10.1016/S0377-2217(99)00447-6. URL http://dx.doi.org/10.1016/S0377-2217(99)00447-6.

Chang Sup Sung and Sang Hum Yoon. Minimizing maximum completion time in a two-batch-processing-machine flowshop with dynamic arrivals allowed. *Engineering Optimization*, 28 (3):231--243, 1997. doi: 10.1080/03052159708941133. URL http://dx.doi.org/10.1080/03052159708941133.

Chang Sup Sung, Young Hwan Kim, and Sang Hum Yoon. A problem reduction and decomposition approach for scheduling for a flowshop of batch processing machines. *European Journal of Operational Research*, 121(1):179--192, 2000. ISSN 0377-2217. doi: 10.1016/S0377-2217(99)00031-4. URL http://dx.doi.org/10.1016/S0377-2217(99)00031-4.

Chang Sup Sung, Y. I. Choung, J. M. Hong, and Young Hwan Kim. Minimizing makespan on a single burn-in oven with job families and dynamic job arrivals. *Computers & Operations Research*, 29(8):995--1007, 2002. ISSN 0305-0548. doi: 10.1016/S0305-0548(00)00098-8.

Raja Sunkara and Ramesh Rao. Use of discrete event simulation to analyze dispatch policies of an equipment group in semiconductor fab. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, WSC '03, pages 1474--1479. ACM Press, 2003. ISBN 0-7803-8132-7. URL http://dl.acm.org/citation.cfm?id=1030818.1031017.

A. N. Swe, Amit Kumar Gupta, Appa Iyer Sivakumar, and Peter Lendermann. Cycle time reduction at cluster tool in semiconductor wafer fabrication. In *Proceedings of the 8th Electronics Packaging Technology Conference (EPTC)*, pages 671--677, 2006.

John Benedict C. Tajan, Appa Iyer Sivakumar, and Stanley B. Gershwin. Online control of a batch processor with incompatible job families under correlated future arrivals. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, pages 2100--2108. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6. URL http://dl.acm.org/citation.cfm?id=1516744.1517111.

John Benedict C. Tajan, Appa Iyer Sivakumar, and Stanley B. Gershwin. Control of a single batch processor with incompatible job families and future job arrivals. *Semiconductor Manufacturing, IEEE Transactions on*, 24(2):208--222, 2011. ISSN 0894-6507. doi: 10.1109/TSM.2011.2120850. URL http://dx.doi.org/10.1109/TSM.2011.2120850.

John Benedict C. Tajan, Appa Iyer Sivakumar, and Stanley B. Gershwin. Heuristic control of multiple batch processors with incompatible job families and future job arrivals. *International Journal of Production Research*, 50(15):4206--4219, 2012. doi: 10.1080/00207543.2011.601342. URL http://dx.doi.org/10.1080/00207543.2011.601342.

El-Ghazali Talbi. *Metaheuristics: From design to implementation*. John Wiley & Sons, Inc., Hoboken and NJ, 2009. ISBN 9780470278581. URL http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10310563.

S. K. Tangudu and Mary E. Kurz. A branch and bound algorithm to minimise total weighted tardiness on a single batch processing machine with ready times and incompatible job families. *Production Planning & Control*, 17(7):728--741, 2006. doi: 10.1080/09537280600901467. URL http://dx.doi.org/10.1080/09537280600901467.

Dusan Teodorovic. Bee colony optimization (bco). In Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri, editors, *Innovations in swarm intelligence*. Springer, Berlin, 2009. ISBN 978-3-642-04224-9.

Vadim G. Timkovsky. Reducibility among scheduling classes. In Joseph Y-T. Leung, editor, *Handbook of scheduling*, pages 165--206. Chapman & Hall/CRC, Boca Raton, 2004. ISBN 9780203489802.

Vincent T'kindt and Jean-Charles Billaut. *Multicriteria Scheduling*. Springer Berlin Heidelberg, [New York], 2 edition, 2006. ISBN 3-540-28230-0.

Robert Unbehaun and Oliver Rose. The use of slow down factors for the analysis and development of scheduling algorithms for parallel cluster tools. In L. Felipe Perrone, Barry Lawson, Jason Liu, and Frederick P. Wieland, editors, *Proceedings of the Winter Simulation Conference*, WSC '06, pages 1840--1847. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

Robert Unbehaun and Oliver Rose. Predicting cluster tool behavior with slow down factors. In Shane G. Henderson, Bahar Biller, Ming-Hua Hsieh, John Shortle, D Tew Jeffrey, and Russell R. Barton, editors, *Proceedings of the 39th Winter Simulation Conference*, WSC '07, pages 1755--1760, Piscataway and NJ and USA, 2007. IEEE Press. ISBN 1-4244-1306-0.

Reha Uzsoy. Scheduling a single batch processing machine with non-identical job sizes. *International Journal of Production Research*, 32(7):1615--1635, 1994. doi: 10.1080/00207549408957026. URL http://dx.doi.org/10.1080/00207549408957026.

Reha Uzsoy. Scheduling batch processing machines with incompatible job families. *International Journal of Production Research*, 33(10):2685--2708, 1995. doi: 10.1080/00207549508904839. URL http://dx.doi.org/10.1080/00207549508904839.

Reha Uzsoy and Yaoyu Yang. Minimizing total weighted completion time on a single batch processing machine. *Production and Operations Management*, 6(1):57--73, 1997. ISSN 1937-5956. doi: 10.1111/j.1937-5956.1997.tb00415.x. URL http://dx.doi.org/10.1111/j.1937-5956.1997.tb00415.x.

Reha Uzsoy, Laura K. Church, Irfan M. Ovacik, and Jim Hinchman. Dispatching rules for semiconductor testing operations: A computational study. In *Electronics Manufacturing Technology Symposium, Thirteenth IEEE/CHMT International*, pages 272 -- 276, 1992a.

Reha Uzsoy, Chung-Yee Lee, and Louis A. Martin-Vega. A review of production planning and scheduling models in the semiconductor industry part i: System characteristics, performance evaluation and production planning. *IIE Transactions*, 24(4):47--60, 1992b. ISSN 0740-817X. doi: 10.1080/07408179208964233. URL http://dx.doi.org/10.1080/07408179208964233.

Reha Uzsoy, Chung-Yee Lee, and Louis A. Martin-Vega. A review of production planning and scheduling models in the semiconductor industry part ii: Shop-floor control. *IIE Transactions*, 26(5):44--55, 1994. ISSN 0740-817X. doi: 10.1080/07408179408966627. URL http://dx.doi.org/10.1080/07408179408966627.

Jorge M. S. Valente. Improving the performance of the atc dispatch rule by using workload data to determine the lookahead parameter value. *International Journal of Production Economics*, 106(2):563--573, 2007. ISSN 0925-5273. doi: 10.1016/j.ijpe.2006.06.017. URL http://dx.doi.org/10.1016/j.ijpe.2006.06.017.

Jannes Remco van den Berg and Edwin Den Hartog. Method and apparatus for batch processing of wafers in a furnace, 2004. URL http://www.freepatentsonline.com/6835039.html.

Roman van der Krogt, James Little, and Helmut Simonis. Scheduling in the real world: Lessons learnt, 2009.

P. van der Meulen. Linear semiconductor manufacturing logistics and the impact on cycle time. In *IEEE/SEMI Advanced Semiconductor Manufacturing Conference 2007 (ASMC 2007)*, pages 111--116, 2007. doi: 10.1109/ASMC.2007.4595693. URL http://dx.doi.org/10.1109/ASMC.2007.4595693.

D. van der Zee. Look-ahead strategies for controlling batch operations in industry-overview, comparison and exploration. In *Simulation Conference, 2000. Proceedings. Winter*, volume 2, pages 1364--1373 vol.2, 2000. doi: 10.1109/WSC.2000.899111. URL http://dx.doi.org/10.1109/WSC.2000.899111.

D. J. van der Zee, A. van Harten, and P. C. Schuur. Dynamic job assignment heuristics for multi-server batch operations- a cost based approach. *International Journal of Production Research*, 35(11):3063--3094, 1997. doi: 10.1080/002075497194291. URL http://dx.doi.org/10.1080/002075497194291.

Durk-Jouke van der Zee. Real-time adaptive control of multi-product multi-server bulk service processes. In *Proceedings of the 33nd conference on Winter simulation*, WSC '01, pages 930--936, Washington and DC and USA, 2001. IEEE Computer Society Press. ISBN 0-7803-7309-X. URL http://dl.acm.org/citation.cfm?id=564124.564256.

Durk-Jouke van der Zee. Adaptive scheduling of batch servers in flow shops. *International Journal of Production Research*, 40(12):2811--2833, 2002. doi: 10.1080/00207540210136559. URL http://dx.doi.org/10.1080/00207540210136559.

Durk-Jouke van der Zee. Look-ahead strategies for controlling batch operations in industry - an overview. In Stephen E. Chick, Paul J. Sanchez, David M. Ferrin, and Douglas J. Morrice, editors, *Proceedings of the 35th Winter Simulation Conference*, volume 2 of *WSC '03*, pages 1480--1487. ACM Press, 2003. ISBN 0-7803-8132-7. doi: 10.1109/WSC.2003.1261592. URL http://dx.doi.org/10.1109/WSC.2003.1261592.

Durk-Jouke van der Zee. Dynamic scheduling of batch servers with compatible product families. *International Journal of Production Research*, 42(22):4803--4826, 2004. doi: 10.1080/00207540412331270450. URL http://dx.doi.org/10.1080/00207540412331270450.

Durk-Jouke van der Zee. Dynamic scheduling of batch-processing machines with non-identical product sizes. *International Journal of Production Research*, 45(10):2327--2349, 2007. doi: 10.1080/00207540600690537. URL http://dx.doi.org/10.1080/00207540600690537.

Durk-Jouke van der Zee, Aart van Harten, and Peter Schuur. On-line scheduling of multi-server batch operations. *IIE Transactions*, 33(7):569--586, 2001. ISSN 0740-817X. doi: 10.1080/07408170108936855. URL http://dx.doi.org/10.1080/07408170108936855.

H. van Dyke Parunak. Book review: Intelligent scheduling by m. zweben and m. s. fox. *SIGART Bull*, 6(2):49--51, 1995. ISSN 0163-5719. doi: 10.1145/201977.1065560. URL http://doi.acm.org/10.1145/201977.1065560.

Peter van Zant. *Microchip fabrication: A practical guide to semiconductor processing*. McGraw-Hill, New York, 5 edition, 2004. ISBN 0071432418.

C. Vasudev. *Combinatorics and graph theory*. New Age International, New Delhi, 2007. ISBN 978-81-224-2890-2.

Mario C. Vélez-Gallego, Purushothaman Damodaran, and Manuel Rodríguez. Makespan minimization on a single batch processing machine with unequal job ready times. *International Journal of Industrial Engineering*, 18(10):536--546, 2011. ISSN 1943670X. URL http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=70244940&site=ehost-live.

Kai Velten. *Mathematical modeling and simulation: Introduction for scientists and engineers*. Wiley-VCH, Weinheim [Germany], 2009. ISBN 3527407588.

Jose A. Ventura. Dynamic programming. In A. Ravi Ravindran, editor, *Operations research and management science handbook*. CRC Press, Boca Raton, 2008. ISBN 978-0849397219.

Ari P. J. Vepsalainen and Thomas E. Morton. Priority rules for job shops with weighted tardiness costs. *Management Science*, 33(8):1035--1047, 1987. doi: 10.1287/mnsc.33.8.1035. URL http://dx.doi.org/10.1287/mnsc.33.8.1035.

Iris F. A. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677--709, 2006. ISSN 0377-2217. doi: 10.1016/j.ejor.2004.09.020. URL http://dx.doi.org/10.1016/j.ejor.2004.09.020.

Christos Voudouris, Edward P. K. Tsang, and Abdullah Alsheddy. Guided local search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Avinash M. Waikar, Bhaba R. Sarker, and Anil M. Lal. A comparative study of some priority dispatching rules under different shop loads. *Production Planning & Control*, 6(4):301--310, 1995. doi: 10.1080/09537289508930284. URL http://dx.doi.org/10.1080/09537289508930284.

Gabriel A. Wainer. *Discrete-event modeling and simulation: A practitioner's approach*. CRC Press, Boca Raton, 2009. ISBN 978-1-4200-5336-4.

Gabriel A. Wainer and Pieter J. Mosterman. *Discrete-event modeling and simulation: Theory and applications*. CRC Press, Boca Raton and FL, 2011. ISBN 978-1-4200-7233-4.

Cheng-Shuo Wang and Reha Uzsoy. A genetic algorithm to minimize maximum lateness on a batch processing machine. *Computers & Operations Research*, 29(12):1621--1640, 2002. ISSN 0305-0548. doi: 10.1016/S0305-0548(01)00031-4. URL http://dx.doi.org/10.1016/S0305-0548(01)00031-4.

Chia-Nan Wang and Chih-Hong Wang. A simulated model for cycle time reduction by acquiring optimal lot size in semiconductor manufacturing. *The International Journal of Advanced Manufacturing Technology*, 34(9-10):1008--1015, 2007. ISSN 0268-3768. doi: 10.1007/s00170-006-0884-9. URL http://dx.doi.org/10.1007/s00170-006-0884-9.

Hui-Mei Wang and Fuh-Der Chou. Solving the parallel batch-processing machines with different release times, job sizes, and capacity limits by metaheuristics. *Expert Systems with Applications*, 37(2):1510--1521, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2009.06.070. URL http://dx.doi.org/10.1016/j.eswa.2009.06.070.

Hui-Mei Wang and Fuh-Der Chou. Bi-criteria single batch-processing machine with job release time and non-identical job sizes. In *International Journal of Information and Education Technology*, volume 5 of *3*, pages 536--539. 2013. URL http://www.ijiet.org/papers/331-K023.pdf.

Xianpeng Wang and Lixin Tang. A population-based variable neighborhood search for the single machine total weighted tardiness problem. *Computers & Operations Research*, 36(6):2105--2110, 2009. ISSN 0305-0548. doi: 10.1016/j.cor.2008.07.009. URL http://dx.doi.org/10.1016/j.cor.2008.07.009.

Jean-Paul Watson. An introduction to fitness landscape analysis and cost models for local search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of metaheuristics*, International Series in Operations Research & Management Science. Springer, New York, 2009. ISBN 978-1-4419-1665-5.

Scott Webster and Kenneth R. Baker. Scheduling groups of jobs on a single machine. *Operations Research*, 43(4):692--703, 1995. doi: 10.1287/opre.43.4.692. URL http://dx.doi.org/10.1287/opre.43.4.692.

Gerald Weigert, Sebastian Werner, and W. Sauer. Adaptive simulation systems for reducing uncertainty of predictions. In *Proceedings of the 9th Conference on Flexible Automation and Intelligent Manufacturing (FAIM)*, pages 691--701, 1999.

Gerald Weigert, Andreas Klemmt, and S. Horn. Design and validation of heuristic algorithms for simulation-based scheduling of a semiconductor backend facility. *International Journal of Production Research*, 47(8):2165--2184, 2009. doi: 10.1080/00207540902744784. URL http://dx.doi.org/10.1080/00207540902744784.

L. M. Wein. Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 1(3):115--130, 1988. ISSN 0894-6507. doi: 10.1109/66.4384. URL http://dx.doi.org/10.1109/66.4384.

L. M. Wein. On the relationship between yield and cycle time in semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 5(2):156--158, 1992. ISSN 0894-6507. doi: 10.1109/66.136277. URL http://dx.doi.org/10.1109/66.136277.

Gerhard Weiss. *Multiagent systems: A modern approach to distributed artificial intelligence*. MIT Press, Cambridge and Mass, 1999. ISBN 9780262731317.

Howard J. Weiss. The computation of optimal control limits for a queue with batch services. *Management Science*, 25(4):320--328, 1979. doi: 10.1287/mnsc.25.4.320. URL http://dx.doi.org/10.1287/mnsc.25.4.320.

Michael Weng. Integer programming. In A. Ravi Ravindran, editor, *Operations research and management science handbook*. CRC Press, Boca Raton, 2008. ISBN 978-0849397219.

W. W. Weng and Robert C. Leachman. An improved methodology for real-time production decisions at batch-process work stations. *Semiconductor Manufacturing, IEEE Transactions on*, 6(3):219--225, 1993. ISSN 0894-6507. doi: 10.1109/66.238169. URL http://dx.doi.org/10.1109/66.238169.

Sebastian Werner and Gerald Weigert. Process accompanying simulation: a general approach for the continuous optimization of manufacturing schedules in electronics production. In Enver Yücesan, C. H. Chen, Jane L. Snowdon, and J. M. Charnes, editors, *Proceedings of the 34th Winter simulation Conference*, WSC '02, pages 1903--1908. Winter Simulation Conference, 2002. ISBN 0-7803-7615-3.

wikipedia. *P versus NP problem*. Wikipedia, The Free Encyclopedia, 2014. URL http://en.wikipedia.org/w/index.php?title=P_versus_NP_problem&oldid=592119550.

Andreas Willig. A short introduction to queueing theory, 1999.

David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on, title=No free lunch theorems for optimization*, 1(1):67--82, 1997. ISSN 1089-778X. doi: 10.1109/4235.585893. URL http://dx.doi.org/10.1109/4235.585893.

Samuel C. Wood. Simple performance models for integrated processing tools. *IEEE Transactions on Semiconductor Manufacturing*, 9(3):320--328, 1996. ISSN 0894-6507.

Samuel C. Wood, Sanjay Tripathi, and Farhad Moghadam. A generic model for cluster tool throughput time and capacity. In *Proceedings of the Advanced Semiconductor Manufacturing Conference and Workshop. (ASMC 94)*, pages 194--199, 1994. doi: 10.1109/ASMC.1994.588245. URL http://dx.doi.org/10.1109/ASMC.1994.588245.

Michael M. Woolfson and Geoffrey J. Pert. *An introduction to computer simulation*. Oxford University Press, Oxford [u.a.], 1999. ISBN 0-19-850423-3.

Szu-Yung David Wu and Richard A. Wysk. An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing. *International Journal of Production Research*, 27(9):1603--1623, 1989. doi: 10.1080/00207548908942642. URL http://dx.doi.org/10.1080/00207548908942642.

Rui Xu, Hua-Ping Chen, George Q. Huang, Hao Shao, Ba-Yi Cheng, and Bo-Wen Liu. Minimizing the total completion time on a single batch processing machine with non-identical job sizes using ant colony optimization. In *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on*, pages 2211--2215, 2008a. doi: 10.1109/ICIEA.2008.4582910. URL http://dx.doi.org/10.1109/ICIEA.2008.4582910.

Rui Xu, Hua-Ping Chen, Jun-Hong Zhu, and Hao Shao. A hybrid ant colony optimization to minimize the total completion time on a single batch processing machine with non-identical job sizes. In *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, volume 7, pages 439--443, 2008b. doi: 10.1109/ICNC.2008.384. URL http://dx.doi.org/10.1109/ICNC.2008.384.

Rui Xu, Huaping Chen, and Xueping Li. Makespan minimization on single batch-processing machine via ant colony optimization. *Computers & Operations Research*, 39(3):582--593, 2012. ISSN 0305-0548. doi: 10.1016/j.cor.2011.05.011. URL http://www.sciencedirect.com/science/article/pii/S0305054811001353.

Rui Xu, Huaping Chen, and Xueping Li. A bi-objective scheduling problem on batch machines via a pareto-based ant colony system. *International Journal of Production Economics*, 145(1):371--386, 2013. ISSN 0925-5273. doi: 10.1016/j.ijpe.2013.04.053. URL http://www.sciencedirect.com/science/article/pii/S0925527313002284.

Shubin Xu and J.C Bean. A genetic algorithm for scheduling parallel non-identical batch processing machines. In *Computational Intelligence in Scheduling, 2007. SCIS '07. IEEE Symposium on*, pages 143--150, 2007. doi: 10.1109/SCIS.2007.367682. URL http://dx.doi.org/10.1109/SCIS.2007.367682.

Shiqing Yao, Zhibin Jiang, and Na Li. A branch and bound algorithm for minimizing total completion time on a single batch machine with incompatible job families and dynamic arrivals. *Computers & Operations Research*, 39(5):939--951, 2012. ISSN 0305-0548. doi: 10.1016/j.cor.2011.06.003. URL http://dx.doi.org/10.1016/j.cor.2011.06.003.

M. Yazdani, M. Amiri, and M. Zandieh. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, 37(1):678--687, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2009.06.007. URL http://dx.doi.org/10.1016/j.eswa.2009.06.007.

Jingang Yi, Shengwei Ding, M.T Zhang, and P. van der Meulen. Throughput analysis of linear cluster tools. In *IEEE International Conference on Automation Science and Engineering (CASE 2007)*, pages 1063--1068, 2007. doi: 10.1109/COASE.2007.4341849. URL http://dx.doi.org/10.1109/COASE.2007.4341849.

Tetsuo Yoshida, Yoshhiaki Sasaki, Saeki Hiroaki, Yasushi Taniyama, and Takizawa Aichi. Port structure in semiconductor processing system, 2007.

Chih-Yuan Yu and Han-Pang Huang. On-line learning delivery decision support system for highly product mixed semiconductor foundry. *Semiconductor Manufacturing, IEEE Transactions on*, 15(2):274--278, 2002. ISSN 0894-6507. doi: 10.1109/66.999604. URL http://dx.doi.org/10.1109/66.999604.

J. J. Yuan, Z. H. Liu, C. T. Ng, and T. C. Edwin Cheng. The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan. *Theoretical Computer Science*, 320(2--3):199--212, 2004. ISSN 0304-3975. doi: 10.1016/j.tcs.2004.01.038. URL http://www.sciencedirect.com/science/article/pii/S0304397504000726.

C. Yugma, Stéphane Dauzère-Pérès, Alexandre Derreumaux, and O. Sibille. A batch optimization solver for diffusion area scheduling in semiconductor manufacturing. In *Advanced Semiconductor Manufacturing Conference, 2008. ASMC 2008. IEEE/SEMI*, pages 327--332, 2008. doi: 10.1109/ASMC.2008.4529063. URL http://dx.doi.org/10.1109/ASMC.2008.4529063.

T. Yurtsever, E. Kutanoglu, and J. Johns. Heuristic based scheduling system for diffusion in semiconductor manufacturing. In M. D. Rossetti, Raymond R. Hill, Björn Johansson, Ann Dunkin, and Ricki G. Ingalls, editors, *Proceedings of the Winter Simulation Conference*, WSC '09, pages 1677--1685, 2009. doi: 10.1109/WSC.2009.5429171. URL http://dx.doi.org/10.1109/WSC.2009.5429171.

Günther Zäpfel, Roland Braune, and Michael Bögl. *Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics*. Springer-Verlag US, 1 edition, 2010. ISBN 3642113427.

Gouchuan Zhang, Xiaoqiang Cai, and C.K Wong. On-line algorithms for minimizing makespan on batch processing machines. *Naval Research Logistics (NRL)*, 48(3):241--258, 2001a. ISSN 1520-6750. doi: 10.1002/nav.5. URL http://dx.doi.org/10.1002/nav.5.

Guochuan Zhang, Xiaoqiang Cai, Chung-Yee Lee, and C.K Wong. Minimizing makespan on a single batch processing machine with nonidentical job sizes. *Naval Research Logistics (NRL)*, 48(3):226--240, 2001b. ISSN 1520-6750. doi: 10.1002/nav.4. URL http://dx.doi.org/10.1002/nav.4.

Huai Zhang, Zhibin Jiang, and Chengtao Guo. Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. *The International Journal of Advanced Manufacturing Technology*, 41(1):110--121, 2009a. ISSN 0268-3768. doi: 10.1007/s00170-008-1462-0. URL http://dx.doi.org/10.1007/s00170-008-1462-0.

Ya-ling Zhang, Hua-Ping Chen, Hao Shao, and Rui Xu. Minimizing makespan for single batch processing machine with non-identical job sizes using a novel algorithm: Free search. In *Information Technology and Computer Science, 2009. ITCS 2009. International Conference on*, volume 1, pages 179--183, 2009b. doi: 10.1109/ITCS.2009.43. URL http://dx.doi.org/10.1109/ITCS.2009.43.

Yuzhong Zhang and Zhigang Cao. An asymptotic ptas for batch scheduling with nonidentical job sizes to minimize makespan. In Andreas Dress, Yinfeng Xu, and Binhai Zhu, editors, *Combinatorial Optimization and Applications*, volume 4616 of *Lecture Notes in Computer Science*, pages 44--51. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73555-7. doi: 10.1007/978-3-540-73556-4_7. URL http://dx.doi.org/10.1007/978-3-540-73556-4_7.

Yuzhong Zhang, Zhigang Cao, and Qingguo Bai. A ptas for scheduling on agreeable unrelated parallel batch processing machines with dynamic job arrivals. In Nimrod Megiddo, Yinfeng Xu,

and Binhai Zhu, editors, *Algorithmic Applications in Management*, volume 3521 of *Lecture Notes in Computer Science*, pages 162--171. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26224-4. doi: 10.1007/11496199_19. URL http://dx.doi.org/10.1007/11496199_19.

Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32--49, 2011. ISSN 2210-6502. doi: 10.1016/j.swevo.2011.03.001. URL http://www.sciencedirect.com/science/article/pii/S2210650211000058.

Shengchao Zhou, Huaping Chen, Rui Xu, and Xueping Li. Minimising makespan on a single batch processing machine with dynamic job arrivals and non-identical job sizes. *International Journal of Production Research*, pages 1--17, 2013. doi: 10.1080/00207543.2013.854937. URL http://dx.doi.org/10.1080/00207543.2013.854937.

Zhugen Zhou and Oliver Rose. A bottleneck detection and dynamic dispatching strategy for semiconductor wafer fabrication facilities. In M. D. Rossetti, Raymond R. Hill, Björn Johansson, Ann Dunkin, and Ricki G. Ingalls, editors, *Proceedings of the Winter Simulation Conference*, WSC '09, pages 1646--1656, 2009. URL http://dl.acm.org/citation.cfm?id=1995456.1995680.

Zhugen Zhou and Oliver Rose. A pull/push concept for toolgroup workload balance in wafer fab. In *Proceedings of the 2010 Winter Simulation Conference (WSC2010)*, pages 2516--2522, 2010. doi: 10.1109/WSC.2010.5678947. URL http://dx.doi.org/10.1109/WSC.2010.5678947.

Zhugen Zhou and Oliver Rose. Wip balance and due date control in a wafer fab with low and high volume products. In *Proceedings of the Winter Simulation Conference*, WSC '12, pages 178:1--178:8. Winter Simulation Conference, 2012. URL http://dl.acm.org/citation.cfm?id=2429759.2429997.

Armin Zimmermann. *Stochastic discrete event systems: Modeling, evaluation, applications*. Springer, Berlin and and New York, 2008. ISBN 9783540741732.

Eckart Zitzler, Lothar Thiele, Kalyanmoy Deb, Carlos A. Coello Coello, and David Corne, editors. *Evolutionary Multi-Criterion Optimization*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41745-3.

Albert Y. Zomaya and Rick Kazman. Simulated annealing techniques. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and theory of computation handbook*. CRC Press, Boca Raton [u.a.], 2010. ISBN 978-1-58488-822-2.

Appendices

A Tabular Literature Overview

machine environment

parallel machines (P)

job release dates (r_j)

yes

incompatible job families ($fmls$)

(no)

non-identical job sizes (B, s_j)

yes

publication	method	objective
$Pm \mid p-batch, b<n, r_j \mid \cdot$		
Li et al. (2004)	A	OTD
Li et al. (2005b)	A	C_{max}
Zhang et al. (2005)	A	C_{max}
Li et al. (2012a)	A#	C_{max}
Koehler and Khuller (2013)	E*	C_{max}
$Pm \mid p-batch, B, s_j, r_j \mid \cdot$		
Chung et al. (2009)	E, H	C_{max}
Chen et al. (2010)	H, MH	C_{max}
Damodaran and Vélez-Gallego (2010)	H	C_{max}
Wang and Chou (2010)	E, MH	C_{max}
Damodaran et al. (2011)	MH	C_{max}
Damodaran and Vélez-Gallego (2012)	MH	C_{max}
Li (2012)	A	C_{max}
Ozturk et al. (2012)	E, E*, A	C_{max}
Xu et al. (2013)	MH	MO
Sahraeian et al. (2014)	RTC	C_{max}

job release dates (r_j)

no

publication	method	objective
$Pm \mid p-batch, b<n \mid \cdot$		
Chandru et al. (1993a)	H	CT
Mönch and Unbehaun (2007)	H*	OTD
$Pm \mid p-batch, B, s_j \mid \cdot$		
Chang et al. (2004)	E, MH	C_{max}
Xu and Bean (2007)	E, MH	C_{max}
Damodaran and Chang (2008)	H	C_{max}
Kashan et al. (2008)	MH	C_{max}
Shao et al. (2008a)	H	C_{max}
Shao et al. (2008b)	H	C_{max}
Chen et al. (2011)	H	C_{max}
Cheng et al. (2012)	E, A	C_{max}, CT
Cheng et al. (2013)	E, MH	C_{max}
Li et al. (2013)	H	C_{max}

single machine (1)

job release dates (r_j)

yes

publication	method	objective
$1 \mid p-batch, b<n, r_j \mid \cdot$		
Ikura and Gimple (1986)	E*	C_{max}
Glassey and Weng (1991)	RTC	CT
Lee et al. (1992)	E*	OTD
Webster and Baker (1995)	E*	OTD, CT
Li and Lee (1997)	E*	OTD
Lee and Uzsoy (1999)	E*, H	C_{max}
Baptiste (2000)	E*	OTD, CT
Liu and Yu (2000)	E*, A	C_{max}
Poon and Zhang (2000)	E*, A	C_{max}
Sung and Choung (2000)	E, H	C_{max}
Zhang et al. (2001a)	A#*, A#	C_{max}
Sung et al. (2002)	E*	C_{max}
Wang and Uzsoy (2002)	MH	OTD
Deng et al. (2003)	E*, A	C_{max}
Gupta et al. (2004)	RTC	OTD
van der Zee (2004)	RTC	CT
Liu and Cheng (2005)	A	CT
Brucker (2007)	E*	OTD, CT
Cao and Yang (2009)	A	MO
Lu et al. (2009b)	E*, A	MO
$1 \mid p-batch, B, s_j, r_j \mid \cdot$		
Chang and Wang (2004)	H	CT
Li et al. (2005a)	A	C_{max}
Chou et al. (2006)	MH	C_{max}
van der Zee (2007)	RTC	CT
Chou and Wang (2008)	E, H, MH	OTD
Lu et al. (2010)	A	C_{max}
Mathirajan et al. (2010)	E, H, MH	OTD
Vélez-Gallego et al. (2011)	E, H	C_{max}
Xu et al. (2012)	E, H, MH	C_{max}
Wang and Chou (2013)	E, MH	MO
Zhou et al. (2013)	H	C_{max}

job release dates (r_j)

no

publication	method	objective
$1 \mid p-batch, b<n \mid \cdot$		
Neuts (1967)	RTC	CT
Lee et al. (1992)	E*	OTD
Albers and Brucker (1993)	E*	CT
Chandru et al. (1993a)	E, H	CT
Webster and Baker (1995)	E*	OTD, CT
Brucker and Kovalyov (1996)	E*, A	OTD
Hochbaum and Landy (1997)	E*, A	CT
Uzsoy and Yang (1997)	E, H	CT
Brucker et al. (1998)	E*	CT
Brucker et al. (1998)	E	C_{max}
Sung and Choung (2000)	E	C_{max}
Chen et al. (2001)	A#	CT
Cai et al. (2002)	A	CT
Deng et al. (2002)	A	CT
Ganesan et al. (2004)	RTC	MO
Poon and Yu (2004)	E	CT
Mönch et al. (2006a)	MH*	OTD
Brucker (2007)	E	C_{max}
Brucker (2007)	E*	OTD, CT
Liu (2007)	E*	OTD
Sabouni and Jolai (2010)	E*	MO
$1 \mid p-batch, B, s_j \mid \cdot$		
Uzsoy (1994)	H	C_{max}
Uzsoy (1994)	E, H	CT
Jolai Ghazvini and Dupont (1998)	H	CT
Azizoglu and Webster (2000)	E	CT
Zhang et al. (2001b)	A	C_{max}
Dupont and Dhaenens-Flipo (2002)	E	C_{max}
Mathirajan et al. (2004)	MH	C_{max}
Melouk et al. (2004)	E, MH	C_{max}
Damodaran et al. (2006)	MH	C_{max}
Kashan et al. (2006a)	MH	C_{max}
Kashan et al. (2006b)	MH	C_{max}
Damodaran et al. (2007)	MH	C_{max}
Zhang and Cao (2007)	A	C_{max}
Erramilli and Mason (2008)	E, MH	OTD
Xu et al. (2008a)	MH	CT
Xu et al. (2008b)	MH	CT
Kashan et al. (2009)	A	C_{max}
Lu et al. (2009a)	MH	MO
Zhang et al. (2009b)	MH	C_{max}
Cheng et al. (2010)	MH	C_{max}
Kashan et al. (2010)	MH	MO
Parsa et al. (2010)	E, H	C_{max}
Malapert et al. (2012)	E	OTD
Jia and Leung (2014)	MH	C_{max}

model: {longest job processing time (L), constant processing time (C), family processing time (F)};

method: {exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*), online setting (#)};

objectives: {makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)};

constraints: {preemption ($prmpt$), rejection ($rjct$), deadlines ($\tilde{d}_j$), machine eligibility (M_j), sequence dependent setup times ($sdst$), reentrant jobs ($rntr$), job splitting ($jspl$), graph compatibility (gc), secondary resources (sr), stochastic processing times (spt), precedence ($prec$), breakdowns ($brkdwn$) }

Literature review of batch scheduling with incompatible job families

machine environment

parallel machines (P)

job release dates (r_j)

yes

incompatible job families ($fmls$) (yes)

$Pm \mid p-batch, b<n, r_j, fmls \mid \cdot$

publication	method	objective
van der Zee et al. (1997)	RTC	CT
Fowler et al. (2000)	RTC	CT
van der Zee (2001)	RTC	CT
van der Zee et al. (2001)	RTC	CT
Cigolini et al. (2002)	RTC	CT
Solomon et al. (2002)	RTC	CT
Habenicht and Mönch (2003)	RTC, MH, H	OTD
Sha et al. (2004)	RTC	OTD
Mönch et al. (2005)	MH	OTD
Mönch et al. (2006b)	H	OTD
Reichelt and Mönch (2006)	MH	MO
Malve and Uzsoy (2007)	MH	OTD
Sha et al. (2007)	RTC	OTD
Li et al. (2008)	MH	OTD
Murray et al. (2008)	RTC	MO
Bar-Noy et al. (2009)	A	OTD
Klemmt et al. (2009)	E, MH	OTD
Li et al. (2009a)	MH	OTD
Chiang et al. (2010)	MH	OTD
Kim et al. (2010)	H	OTD
Tajan et al. (2012)	RTC	CT
Chang et al. (2013)	MH	OTD

$Pm \mid p-batch, b<n, fmls \mid \cdot$

publication	method	objective
Uzsoy (1995)	H	C_{max} , OTD, CT
Habenicht and Mönch (2003)	RTC	OTD
Balasubramanian et al. (2004)	H, MH	OTD
Raghavan and Venkataramana (2006)	MH	OTD
Mönch and Almeder (2009)	MH	OTD
Almeder and Mönch (2011)	MH	OTD
Li et al. (2012b)	A#*	OTD

$Pm \mid p-batch, B, s_j, r_j, fmls \mid \cdot$

publication	method	objective
Mathirajan and Sivakumar (2006a)	H	OTD
Klemmt et al. (2008)	E, H	C_{max} , CT
Yugma et al. (2008)	MH	MO
Li et al. (2009b)	MH	MO
Klemmt et al. (2011)	E, H	C_{max} , OTD, CT
Kohn and Rose (2012)	MH	OTD, CT
Cakici et al. (2013)	E, MH	CT
Kohn and Rose (2013)	MH	CT
Kohn et al. (2013)	MH	MO
Gokhale and Mathirajan (2014)	E, H	OTD

$Pm \mid p-batch, B, s_j, fmls \mid \cdot$

publication	method	objective
Koh et al. (2004)	H, MH	C_{max} , CT
Payman and Leachman (2010)	H	MO

job release dates (r_j)

yes

$1 \mid p-batch, b<n, r_j, fmls \mid \cdot$

publication	method	objective
Fowler et al. (1992a)	RTC	CT
Fowler et al. (1992b)	RTC	CT
Weng and Leachman (1993)	RTC	CT
Robinson et al. (1995)	RTC	CT
Uzsoy (1995)	E	C_{max}
Uzsoy (1995)	H	OTD
Duenyas and Neale (1997)	RTC	CT
Boudhar (2003)	E*, H	C_{max}
Korkmaz (2004)	E, H	CT
Gupta and Sivakumar (2006)	RTC	OTD, MO
Tangudu and Kurz (2006)	E	OTD
Kurz and Mason (2008)	H	OTD
Li and Qiao (2008)	MH	OTD
Nong et al. (2008a)	A	C_{max}
Tajan et al. (2008)	RTC	CT
Guo et al. (2010)	MH	OTD
Tajan et al. (2011)	RTC, E	CT
Yao et al. (2012)	E	CT
Jia et al. (2013)	H	OTD, CT

$1 \mid p-batch, b<n, fmls \mid \cdot$

publication	method	objective
Chandru et al. (1993b)	E*	CT
Uzsoy (1995)	E	C_{max} , OTD, CT
Duenyas and Neale (1997)	RTC	CT
Kim et al. (1998)	RTC	MO
Mehta and Uzsoy (1998)	E*, H	OTD
Akcali et al. (2000)	RTC	CT
Devpura et al. (2001)	H	OTD
Kim et al. (2001)	RTC	OTD
Boudhar (2003)	E*, H	C_{max}
Jolai (2005)	E*	OTD
Perez et al. (2005)	H	OTD
Finke et al. (2008)	E*	C_{max}
Liu and Zhang (2008)	E	OTD
Nong et al. (2008b)	A#	C_{max}
Sabouni and Jolai (2010)	E*, H	MO
Meng and Lu (2011)	A#	C_{max}
Dauzère-Pérès and Mönch (2013)	E, MH	OTD

$1 \mid p-batch, B, s_j, r_j, fmls \mid \cdot$

publication	method	objective
Nong et al. (2008a)	A	C_{max}
Gokhale and Mathirajan (2011)	E, H	OTD

$1 \mid p-batch, B, s_j, fmls \mid \cdot$

publication	method	objective
Hoitomt and Luh (1992)	H	OTD
Kempf et al. (1998)	E*, H	C_{max} , CT
Azizoglu and Webster (2001)	E	CT
Dobson and Nambimadom (1992)	E, E*, H	CT
Dobson and Nambimadom (2001)	E, E*, H	CT
Koh et al. (2005)	E, H, MH	C_{max} , CT
Kashan and Karimi (2007)	MH	OTD

no

non-identical job sizes (B, s_j)

yes

model:{longest job processing time (L), constant processing time (C), family processing time (F)};

method: {exact method (E), heuristic (H), real-time control (RTC), metaheuristic (MH), approximation algorithm (A), special case(s) (*), online setting (#)};

objectives:{makespan (C_{max}), cycle time (CT), on-time delivery (OTD), multiple objectives (MO)};

constraints:{preemption ($prmpt$), rejection ($rjct$), deadlines (d_j), machine eligibility (M_j), sequence dependent setup times ($sdst$), reentrant jobs ($rntr$), job splitting ($jspl$), graph compatibility (gc), secondary resources (sr), stochastic processing times (spt), precedence ($prec$), breakdowns ($brkdwn$) }

B Complexities of Batch Scheduling Problems

Complexities of Batch Scheduling Problems

machine environment

parallel machines (P)

job release dates (r_j)

yes

incompatible job families ($fmls$)

no

$Pm \mid p-batch, b<n, r_j \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$Pm \mid p-batch, b<n \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum T_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	open	(Brucker and Knust, 2014)
$\sum w_j C_j$	NP-hard	$Pm \mid\mid \sum w_j C_j$ (Brucker and Knust, 2014)

$Pm \mid p-batch, B, s_j, r_j \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$Pm \mid p-batch, B, s_j \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum T_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid p-batch, B, s_j \mid \sum C_j$ (Uzsoy, 1994)
$\sum w_j C_j$	NP-hard	$Pm \mid\mid \sum w_j C_j$ (Brucker and Knust, 2014)

yes

identical job sizes (B)

no

$Pm \mid p-batch, b<n, r_j, fmls \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$Pm \mid p-batch, b<n, fmls \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum T_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	open	(Brucker and Knust, 2014)
$\sum w_j C_j$	NP-hard	$Pm \mid\mid \sum w_j C_j$ (Brucker and Knust, 2014)

$Pm \mid p-batch, B, s_j, r_j, fmls \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$Pm \mid p-batch, B, s_j, fmls \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
L_{max}	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j U_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum T_j$	NP-hard	$Pm \mid\mid C_{max}$ (Garey and Johnson, 1978)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid p-batch, B, s_j \mid \sum C_j$ (Uzsoy, 1994)
$\sum w_j C_j$	NP-hard	$Pm \mid\mid \sum w_j C_j$ (Brucker and Knust, 2014)

single machine (1)

job release dates (r_j)

yes

incompatible job families ($fmls$)

no

$1 \mid p-batch, b<n, r_j \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	(Brucker et al., 1998)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$1 \mid p-batch, b<n \mid \cdot$

objective	complexity	proof
C_{max}	Polynomial	(Brucker et al., 1998)
L_{max}	NP-hard	(Brucker et al., 1998)
$\sum U_j$	NP-hard	$1 \mid p-batch, b<n \mid L_{max}$ (Brucker et al., 1998)
$\sum w_j U_j$	NP-hard	$1 \mid p-batch, b<n \mid L_{max}$ (Brucker et al., 1998)
$\sum T_j$	NP-hard	$1 \mid p-batch, b<n \mid L_{max}$ (Brucker et al., 1998)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	open	(Brucker and Knust, 2014)
$\sum w_j C_j$	open	(Brucker and Knust, 2014)

$1 \mid p-batch, B, s_j, r_j \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$1 \mid p-batch, B, s_j \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	(Uzsoy, 1994)
L_{max}	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum U_j$	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum w_j U_j$	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum T_j$	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	(Uzsoy, 1994)
$\sum w_j C_j$	NP-hard	$1 \mid p-batch, B, s_j \mid \sum C_j$ (Uzsoy, 1994)

yes

identical job sizes (B)

no

$1 \mid p-batch, b<n, r_j, fmls \mid \cdot$

objective	complexity	proof
C_{max}	Polynomial	(Uzsoy, 1995)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$1 \mid p-batch, b<n, fmls \mid \cdot$

objective	complexity	proof
C_{max}	Polynomial	(Uzsoy, 1995)
L_{max}	Polynomial	(Uzsoy, 1995)
$\sum U_j$	NP-hard	(Liu and Zhang, 2008)
$\sum w_j U_j$	NP-hard	$1 \mid p-batch, b<n, fmls \mid \sum U_j$ (Liu and Zhang, 2008)
$\sum T_j$	NP-hard	(Mehta and Uzsoy, 1998)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	Polynomial	(Uzsoy, 1995)
$\sum w_j C_j$	Polynomial	(Uzsoy, 1995)

$1 \mid p-batch, B, s_j, r_j, fmls \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
L_{max}	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j U_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum T_j$	NP-hard	$1 \mid r_j \mid L_{max}$ (Lenstra et al., 1977)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)
$\sum w_j C_j$	NP-hard	$1 \mid r_j \mid \sum C_j$ (Lenstra et al., 1977)

$1 \mid p-batch, B, s_j, fmls \mid \cdot$

objective	complexity	proof
C_{max}	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
L_{max}	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum U_j$	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum w_j U_j$	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum T_j$	NP-hard	$1 \mid p-batch, B, s_j \mid C_{max}$ (Uzsoy, 1994)
$\sum w_j T_j$	NP-hard	$1 \mid\mid \sum w_j T_j$ (Lawler, 1977)
$\sum C_j$	NP-hard	$1 \mid p-batch, B, s_j \mid \sum C_j$ (Uzsoy, 1994)
$\sum w_j C_j$	NP-hard	$1 \mid p-batch, B, s_j \mid \sum C_j$ (Uzsoy, 1994)

yes

identical job sizes (B, s_j)

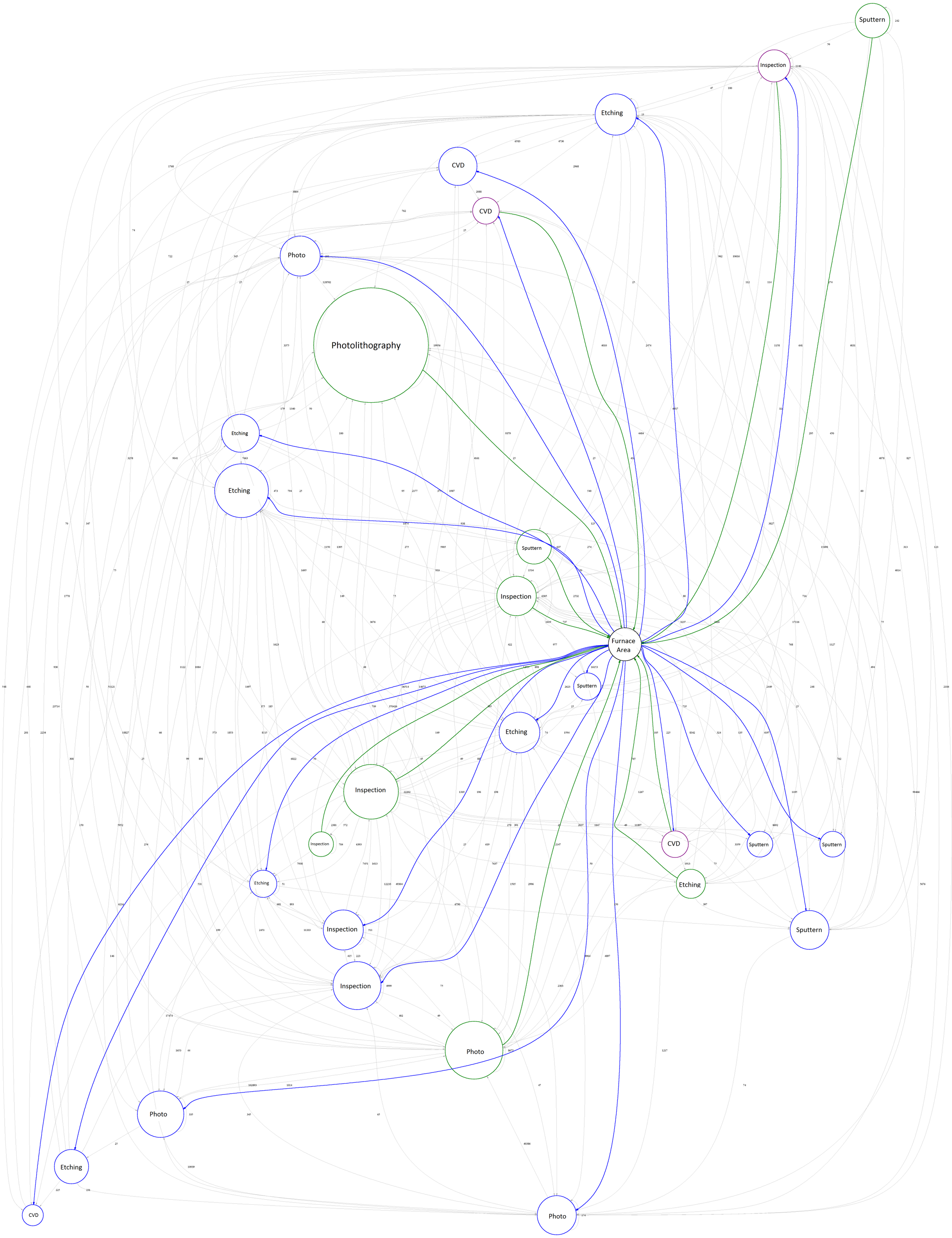
NP-hard stands for NP-hard in the strong case for all cases, i.e. NP-hardness proven under the unary encoding scheme

C Run Times for Exact Methods

Reported Run times of Exact Algorithms Solving Batch Scheduling Problems

size		constraints					run time (min)				
machines	jobs	B, s_j	r_j	$fmls$	others	objective	min	max	method	CPU/RAM	source
1	10	✓	-	-	-	$\sum C_j$	$<< 1$		BnB	IBM PS/6000	Azizoglu and Webster (2000)
1	10	✓	-	-	-	$\sum w_j C_j$	$<< 1$		BnB	IBM PS/6000	Azizoglu and Webster (2000)
1	10	-	✓	✓	-	C_{max}	$<< 1$		DP	IBM Pentium 200MHz	Sung et al. (2002)
1	10	✓	-	-	-	C_{max}	< 1	< 11	CPLEX	Pentium IV 2.2GHz / 512MB RAM	Melouk et al. (2004)
1	10	✓	-	-	-	C_{max}	< 1	< 11	CPLEX	Pentium III 500MHz / 261MB RAM	Damodaran et al. (2006)
1	10	✓	✓	-	-	C_{max}	$<< 1$	< 1	CPLEX	Pentium IV 2.99 GHz / 3GB RAM	Xu et al. (2012)
1	10	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
1	15	✓	-	-	-	$\sum C_j$	$<< 1$		BnB	IBM PS/6000	Azizoglu and Webster (2000)
1	15	✓	-	-	-	$\sum w_j C_j$	$<< 1$		BnB	IBM PS/6000	Azizoglu and Webster (2000)
1	15	✓	-	✓	-	$\sum w_j C_j$	$<< 1$		BnB	IBM PS/6000	Azizoglu and Webster (2001)
1	15	✓	✓	-	-	$\sum w_j T_j$	> 166	> 300	CPLEX	Power Mac G5 / 2GB RAM	Chou and Wang (2008)
1	15	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
1	18	-	✓	✓	-	$\sum C_j$	$<< 1$		BnB	Intel Core2 Duo 2.0 GHz 2GB RAM	Yao et al. (2012)
1	20	✓	-	-	-	$\sum w_j C_j$	< 1		BnB	IBM PS/6000	Azizoglu and Webster (2000)
1	20	✓	-	✓	-	$\sum w_j C_j$	$<< 1$	< 1	BnB	IBM PS/6000	Azizoglu and Webster (2001)
1	20	-	✓	✓	-	C_{max}	$<< 1$	< 1	DP	IBM Pentium 200MHz	Sung et al. (2002)
1	20	✓	-	-	-	C_{max}	> 30	> 30	CPLEX	Pentium IV 2.2GHz / 512MB RAM	Melouk et al. (2004)
1	20	✓	-	-	-	C_{max}	> 30	> 30	CPLEX	Pentium III 500MHz / 261MB RAM	Damodaran et al. (2006)
1	20	✓	-	-	-	C_{max}	> 3	> 30	CPLEX	Pentium IV 1.89GHz / 256MB RAM	Damodaran et al. (2007)
1	20	✓	-	-	-	C_{max}	$<< 1$	< 1	BnP	Pentium IV 2.4GHz / 512MB RAM	Parsa et al. (2010)
1	20	✓	✓	-	-	C_{max}	> 5	> 30	CPLEX	Pentium IV 2.99 GHz / 3GB RAM	Xu et al. (2012)
1	20	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
1	24	-	✓	✓	-	$\sum C_j$	$<< 1$	< 1	BnB	Intel Core2 Duo 2.0 GHz 2GB RAM	Yao et al. (2012)
1	25	✓	-	-	-	$\sum C_j$	< 1	> 30	BnB	IBM PS/6000	Azizoglu and Webster (2000)
1	25	✓	-	-	-	$\sum w_j C_j$	> 3	> 30	BnB	IBM PS/6000	Azizoglu and Webster (2000)
1	25	✓	-	✓	-	$\sum w_j C_j$	$<< 1$	< 2	BnB	IBM PS/6000	Azizoglu and Webster (2001)
1	25	✓	✓		$p_j = p$	C_{max}	< 1	< 1	CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
1	30	✓	-	✓	-	$\sum w_j C_j$	$<< 1$	> 30	BnB	IBM PS/6000	Azizoglu and Webster (2001)
1	30	-	✓	✓	-	C_{max}	$<< 1$	> 15	DP	IBM Pentium 200MHz	Sung et al. (2002)
1	40	✓	-	-	-	C_{max}	< 1	> 30	BnB	Pentium II 266MHz	Dupont and Dhaenens-Flipo (2002)
1	40	-	✓	✓	-	C_{max}	$<< 1$	> 15	DP	IBM Pentium 200MHz	Sung et al. (2002)
1	40	✓	-	-	-	C_{max}	$<< 1$	> 30	BnP	Pentium IV 2.4GHz / 512MB RAM	Parsa et al. (2010)
1	40	-	✓	✓	-	$\sum C_j$	< 1	< 12	BnB	Intel Core2 Duo 2.0 GHz 2GB RAM	Yao et al. (2012)
1	50	✓	-	-	-	C_{max}	> 30		CPLEX	Pentium IV 2.2GHz / 512MB RAM	Melouk et al. (2004)
1	50	✓	-	-	-	C_{max}	> 30		CPLEX	Pentium III 500MHz / 261MB RAM	Damodaran et al. (2006)
1	50	✓	-	-	-	C_{max}	> 30		CPLEX	Pentium IV 1.89GHz / 256MB RAM	Damodaran et al. (2007)
1	50	✓	✓	-	-	$\sum w_j T_j$	> 120		CPLEX	Power Mac G5 / 2GB RAM	Chou and Wang (2008)
1	50	✓	✓	-	-	C_{max}	> 30		CPLEX	Pentium IV 2.99 GHz / 3GB RAM	Xu et al. (2012)
1	60	✓	-	-	-	C_{max}	< 1	> 30	BnB	Pentium II 266MHz	Dupont and Dhaenens-Flipo (2002)
1	60	✓	-	-	-	C_{max}	$<< 1$	> 30	BnP	Pentium IV 2.4GHz / 512MB RAM	Parsa et al. (2010)
1	60	-	✓	✓	-	$\sum C_j$	> 5	< 28	BnB	Intel Core2 Duo 2.0 GHz 2GB RAM	Yao et al. (2012)
1	75	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium IV 1.89GHz / 256MB RAM	Damodaran et al. (2007)
1	80	✓	-	-	-	C_{max}	< 1	> 30	BnB	Pentium II 266MHz	Dupont and Dhaenens-Flipo (2002)
1	80	✓	-	-	-	C_{max}	< 1	> 30	BnP	Pentium IV 2.4GHz / 512MB RAM	Parsa et al. (2010)
1	100	✓	-	-	-	C_{max}	< 1	> 30	BnB	Pentium II 266MHz	Dupont and Dhaenens-Flipo (2002)
1	100	✓	-	-	-	C_{max}	> 300		CPLEX	Pentium IV 2.2GHz / 512MB RAM	Melouk et al. (2004)
1	100	✓	-	-	-	C_{max}	> 300		CPLEX	Pentium III 500MHz / 261MB RAM	Damodaran et al. (2006)
1	100	✓	✓	-	-	$\sum w_j T_j$	> 120		CPLEX	Power Mac G5 / 2GB RAM	Chou and Wang (2008)
1	100	✓	-	-	-	C_{max}	< 1	> 30	BnP	Pentium IV 2.4GHz / 512MB RAM	Parsa et al. (2010)
1	100	✓	✓	-	-	C_{max}	> 30		CPLEX	Pentium IV 2.99 GHz / 3GB RAM	Xu et al. (2012)
2	7	✓	✓	-	-	C_{max}	< 1	< 71	CPLEX	Pentium IV 3.2GHz PC	Chung et al. (2009)
2	10	✓	-	-	-	C_{max}	< 1	> 60	CPLEX	Pentium IV 2.2-GHz / 512 MB RAM	Chang et al. (2004)
2	10	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
2	15	✓	-	-	-	C_{max}	< 1	> 60	CPLEX	Pentium III 997 MHz / 512MB RAM	Xu and Bean (2007)
2	15	✓	✓	-	-	C_{max}	> 4	> 480	CPLEX	Pentium IV 3.2GHz PC	Chung et al. (2009)
2	15	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
2	20	✓	✓	-	-	C_{max}	> 300		CPLEX	Power Mac G5 / 2GB RAM	Wang and Chou (2010)
2	20	✓	✓	-	$p_j = p$	C_{max}	< 5		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
2	25	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium IV 2.2-GHz / 512 MB RAM	Chang et al. (2004)
2	25	✓	✓	-	$p_j = p$	C_{max}	< 10		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
2	50	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium IV 2.2-GHz / 512 MB RAM	Chang et al. (2004)
2	50	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium III 997 MHz / 512MB RAM	Xu and Bean (2007)
2	100	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium III 997 MHz / 512MB RAM	Xu and Bean (2007)
2	100	✓	✓	-	-	C_{max}	> 600		CPLEX	Power Mac G5 / 2GB RAM	Wang and Chou (2010)
3	7	✓	✓	-	-	C_{max}	< 1	< 723	CPLEX	Pentium IV 3.2GHz PC	Chung et al. (2009)
3	10	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
3	15	✓	✓	-	-	C_{max}	> 13	> 480	CPLEX	Pentium IV 3.2GHz PC	Chung et al. (2009)
3	15	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
3	20	✓	✓	-	$p_j = p$	C_{max}	< 8		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
3	25	✓	✓	-	$p_j = p$	C_{max}	< 20		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
4	10	✓		-	-	C_{max}	< 1	> 60	CPLEX	Pentium IV 2.2-GHz / 512 MB RAM	Chang et al. (2004)
4	10	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
4	15	✓	-	-	-	C_{max}	< 1	> 60	CPLEX	Pentium III 997 MHz / 512MB RAM	Xu and Bean (2007)
4	15	✓	✓	-	$p_j = p$	C_{max}	< 1		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
4	20	✓	✓	-	-	C_{max}	< 1	> 300	CPLEX	Power Mac G5 / 2GB RAM	Wang and Chou (2010)
4	20	✓	✓	-	$p_j = p$	C_{max}	> 5	< 6	CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
4	25	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium IV 2.2-GHz / 512 MB RAM	Chang et al. (2004)
4	25	✓	✓	✓	M_j, $\bar{d}_j$	$\sum C_j$	660		CPLEX	-	Klemmt et al. (2008)
4	25	✓	✓	-	$p_j = p$	C_{max}	< 16		CPLEX	Intel Core2 Duo 3Ghz / 3.25 GB RAM	Ozturk et al. (2012)
4	50	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium IV 2.2-GHz / 512 MB RAM	Chang et al. (2004)
4	50	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium III 997 MHz / 512MB RAM	Xu and Bean (2007)
4	100	✓	-	-	-	C_{max}	> 60		CPLEX	Pentium III 997 MHz / 512MB RAM	Xu and Bean (2007)
4	100	✓	✓	-	-	C_{max}	> 600		CPLEX	Power Mac G5 / 2GB RAM	Wang and Chou (2010)
5	30	✓	✓	✓	M_j, $\bar{d}_j$	C_{max}	300		CPLEX	-	Klemmt et al. (2008)
5	30	✓	✓	✓	M_j, $\bar{d}_j$	$\sum C_j$	> 1440		CPLEX	-	Klemmt et al. (2008)
6	20	✓	✓	-	-	C_{max}	< 1	> 300	CPLEX	Power Mac G5 / 2GB RAM	Wang and Chou (2010)
6	40	✓	✓	✓	M_j, $\bar{d}_j$	C_{max}			CPLEX	-	Klemmt et al. (2008)
6	40	✓	✓	✓	M_j, $\bar{d}_j$	$\sum C_j$	> 1440		CPLEX	-	Klemmt et al. (2008)
6	100	✓	✓	-	-	C_{max}	> 600		CPLEX	Power Mac G5 / 2GB RAM	Wang and Chou (2010)
8	60	✓	✓	✓	M_j, $\bar{d}_j$	$\sum C_j$	> 1440		CPLEX	-	Klemmt et al. (2008)
8	60	✓	✓	✓	M_j, $\bar{d}_j$	C_{max}	30		CPLEX	-	Klemmt et al. (2008)

D Material Flow



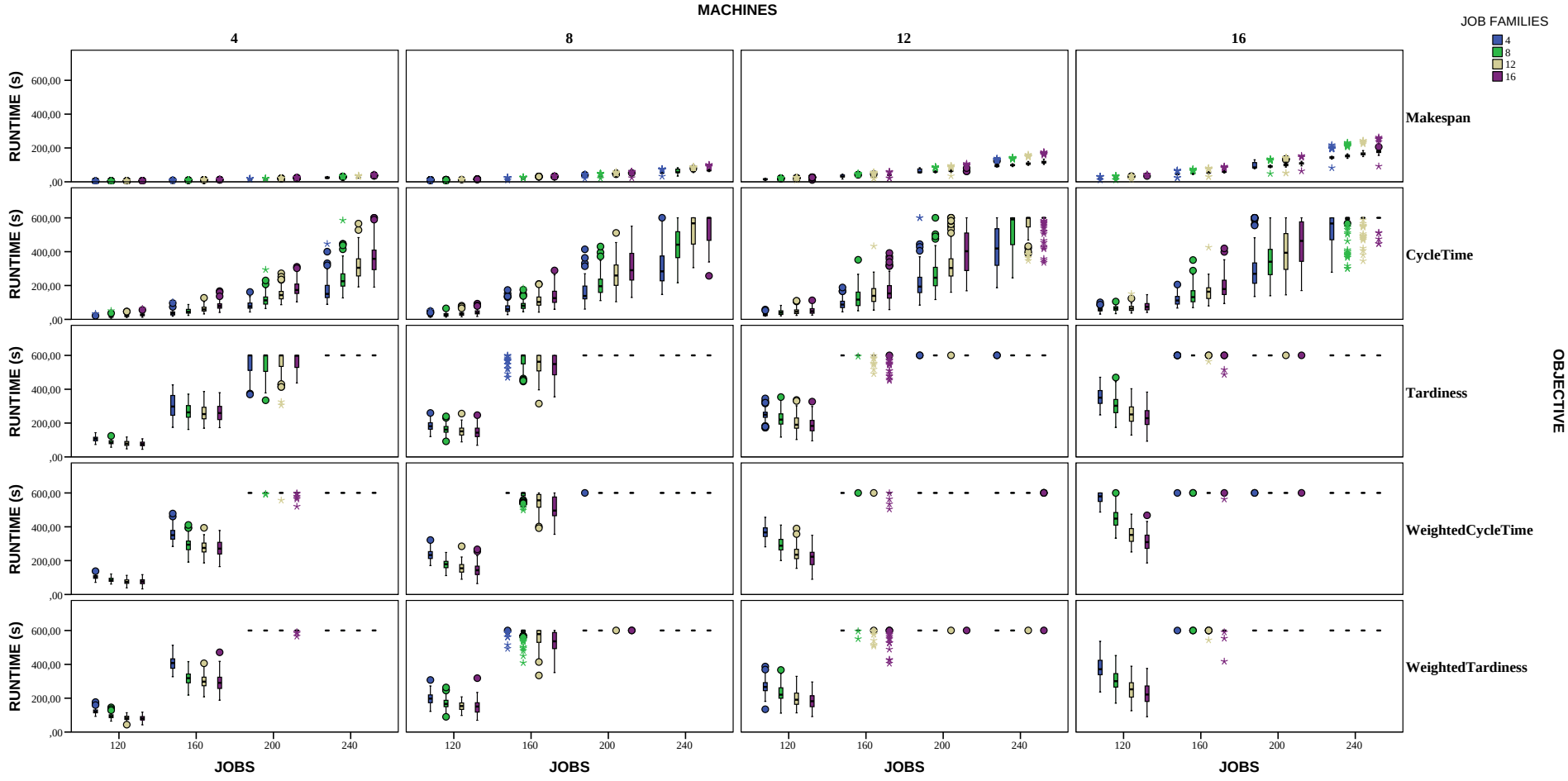
Visualization of the material flow around a typical furnace workcenter

E Experiments

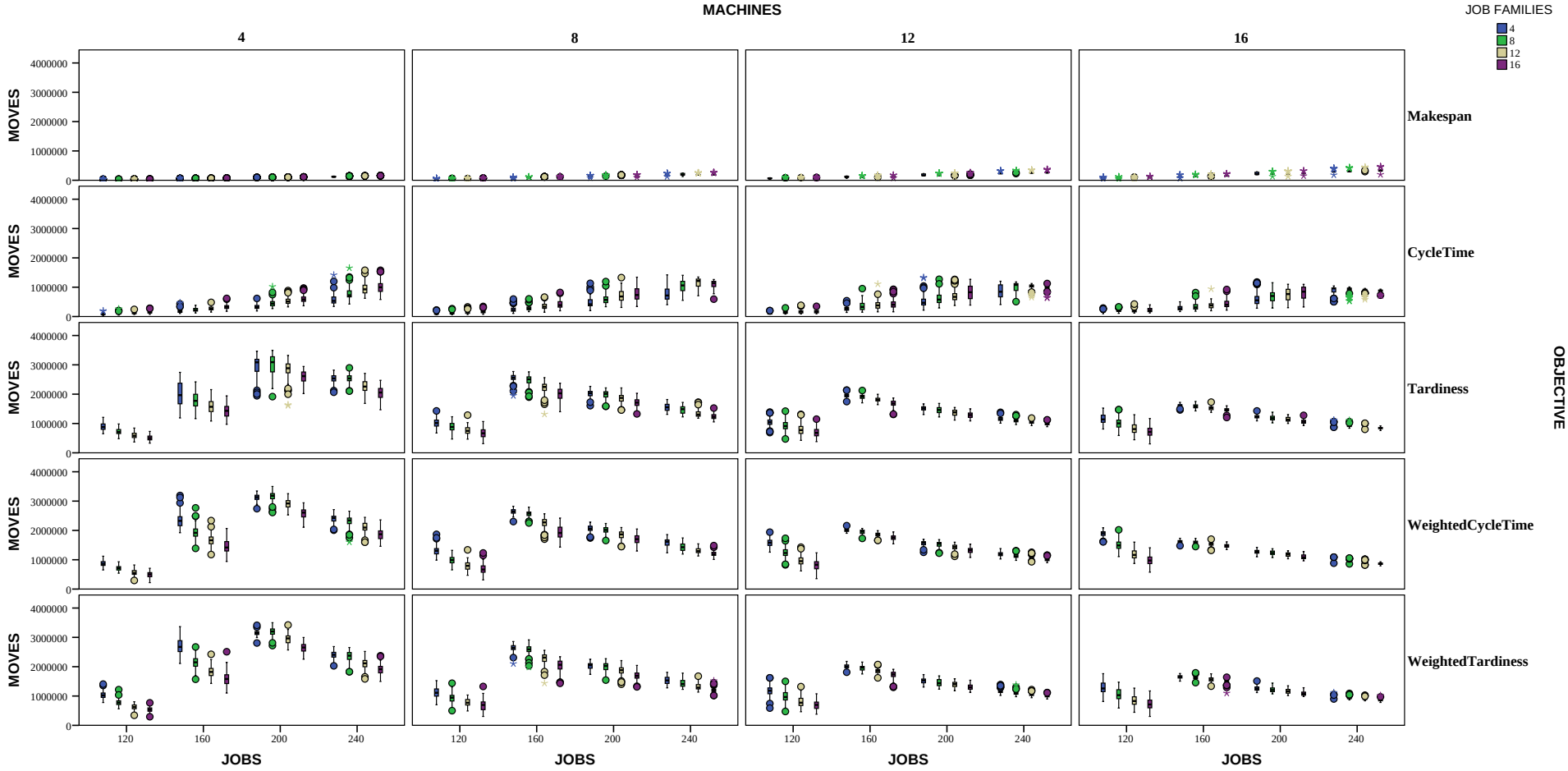
E.1 Experiment E.1

Design of experiment E.1

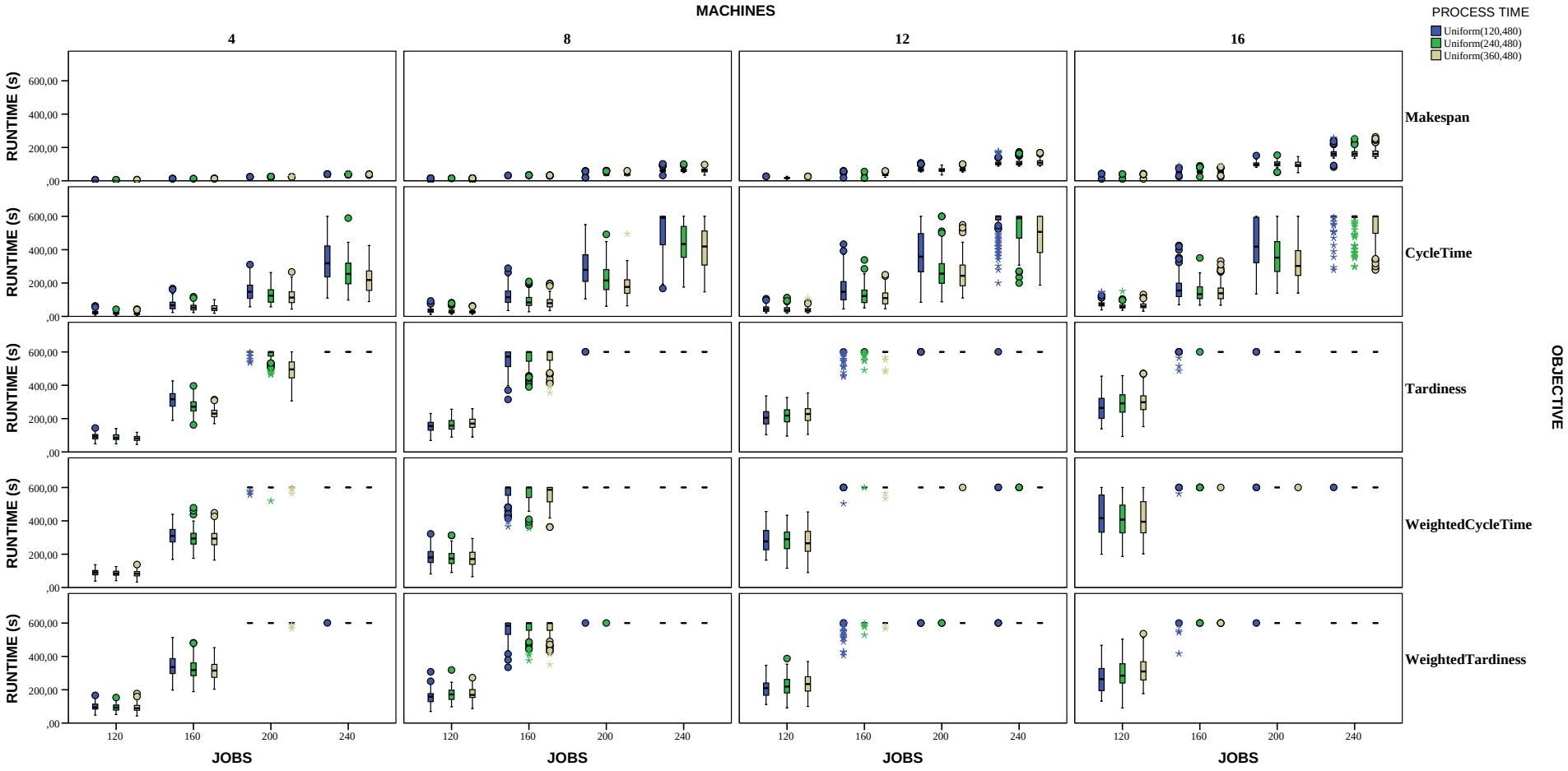
factor	level	count
machines	4, 8, 12, 16	4
jobs	120, 160, 200, 240	4
job families	4, 8, 12, 16	4
dedication density	$1(uniform)$	1
processing times	$U(120,480), U(240,480), U(360,480)$	3
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	$w_j \in U(1,5)$	1
job due dates	$d_j \in r_j + N(12,12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		192
model instances per parameter combination		30
total number of model instances		5760
initial solution	random	1
objective(s)	C_{max}, TCT, TT, TWCT, TWT	5
VNS type	VND	1
deadline	10 min	1
number of methods and settings		5
replications per run		1
total number of runs		28800



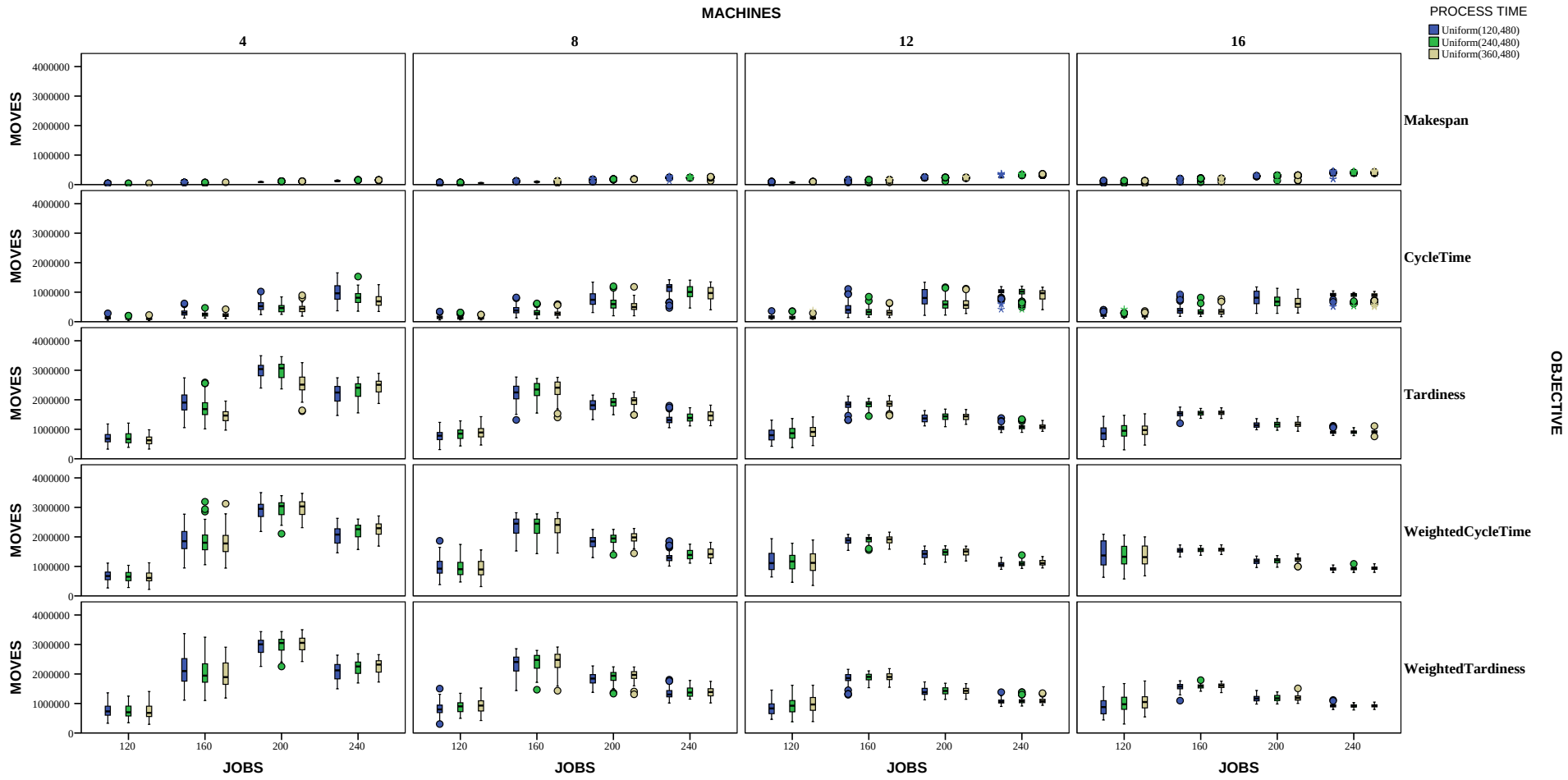
Experiment E1: run times for objectives depending on basic model settings (view 1)



Experiment E1: moves for objectives depending on basic model settings (view 1)



Experiment E1: run times for objectives depending on basic model settings (view 2)

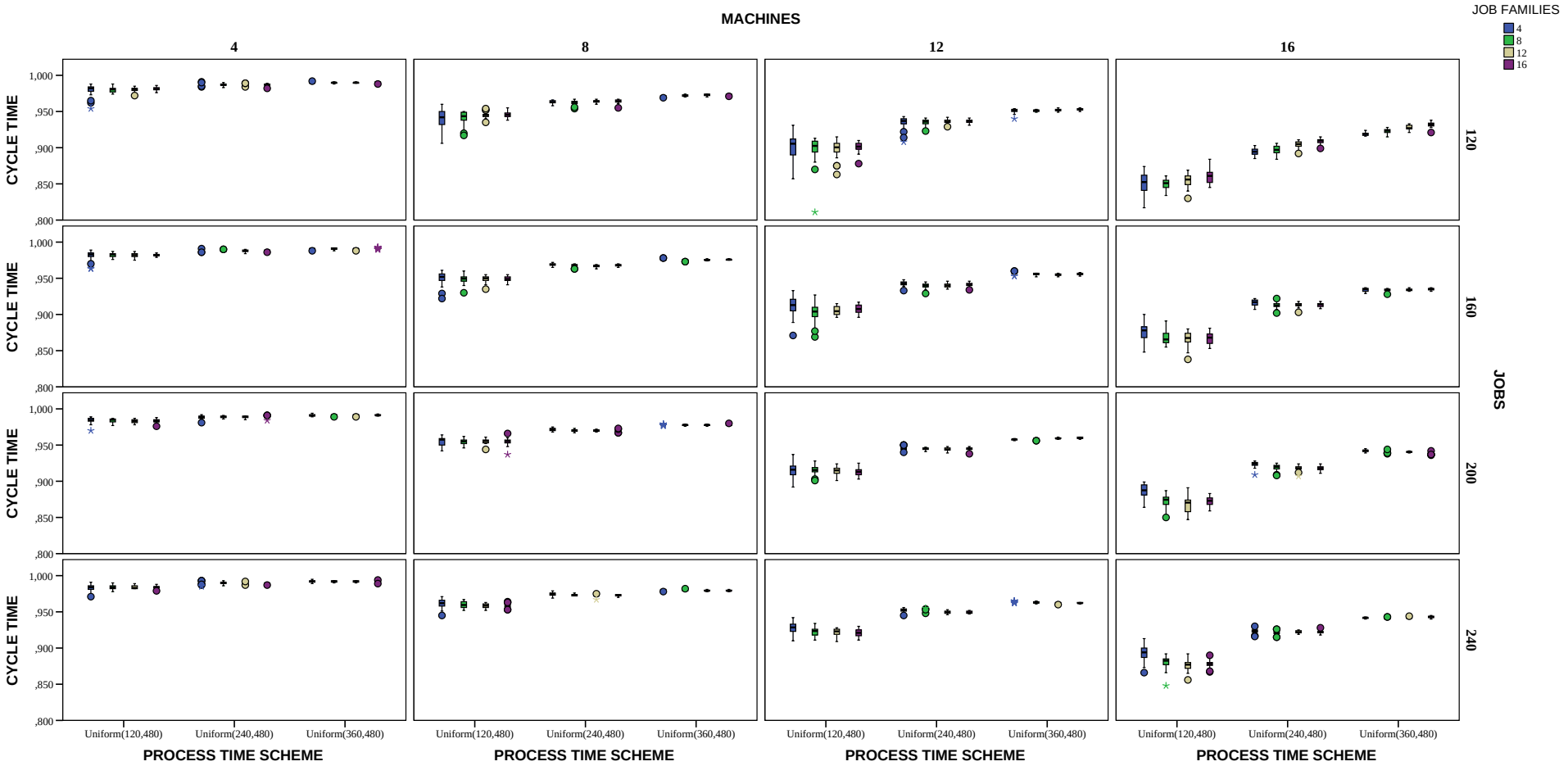


Experiment E1: moves for objectives depending on basic model settings (view 2)

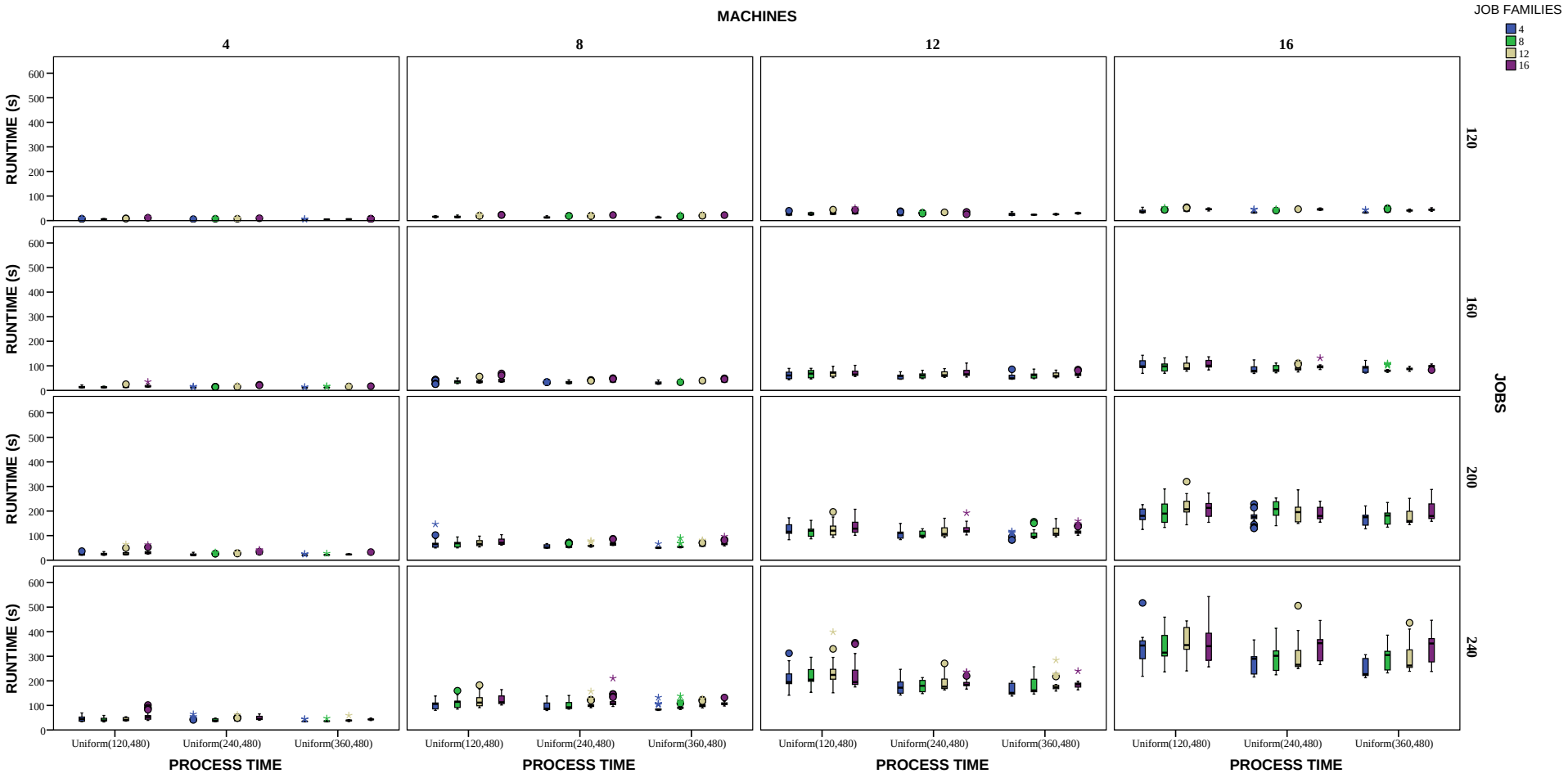
E.2 Experiment E.2

Design of experiment E.2

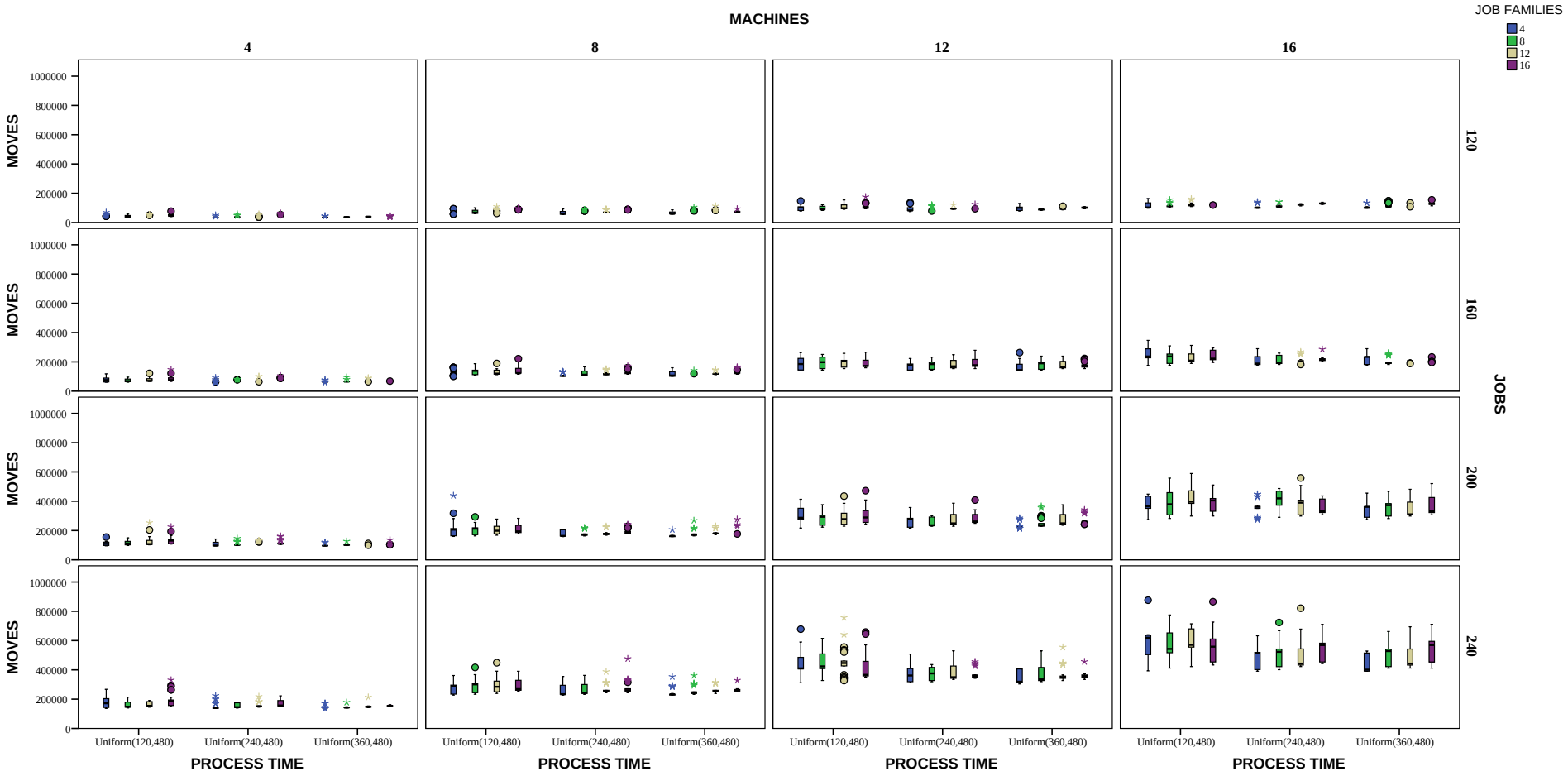
factor	level	count
machines	4, 8, 12, 16	4
jobs	120, 160, 200, 240	4
job families	4, 8, 12, 16	4
dedication density	$1(uniform)$	1
processing times	$U(120, 480), U(240, 480), U(360, 480)$	3
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	$d_j \in r_j + N(12, 12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		192
model instances per parameter combination		30
total number of model instances		5760
initial solution	EDD (TT) / SPT (TCT)	1
objective(s)	TCT, TT	2
VNS type	VND	1
deadline	10 min	1
number of methods and settings		2
replications per run		1
total number of runs		11520



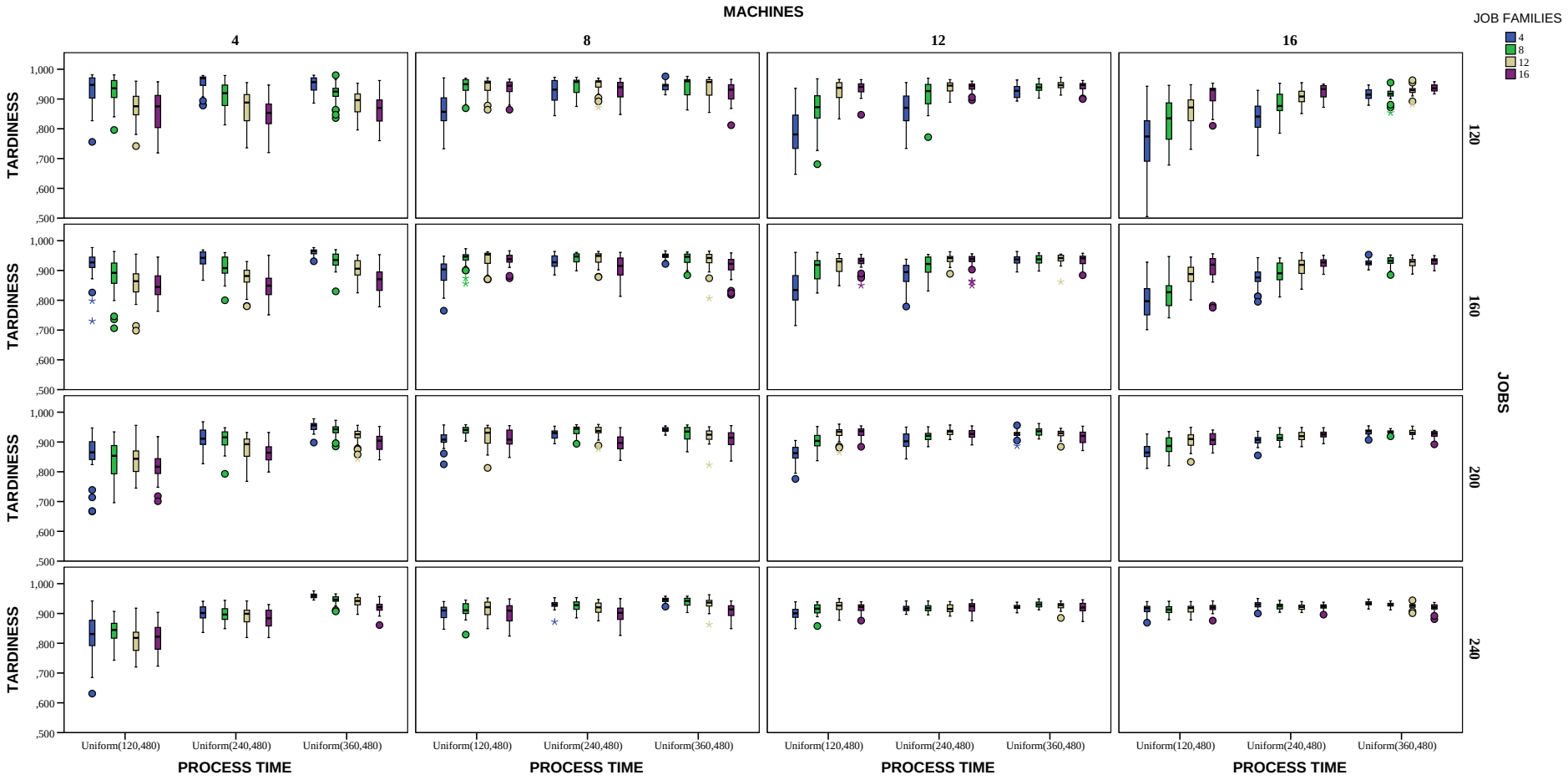
Experiment E2: cycle time improvements



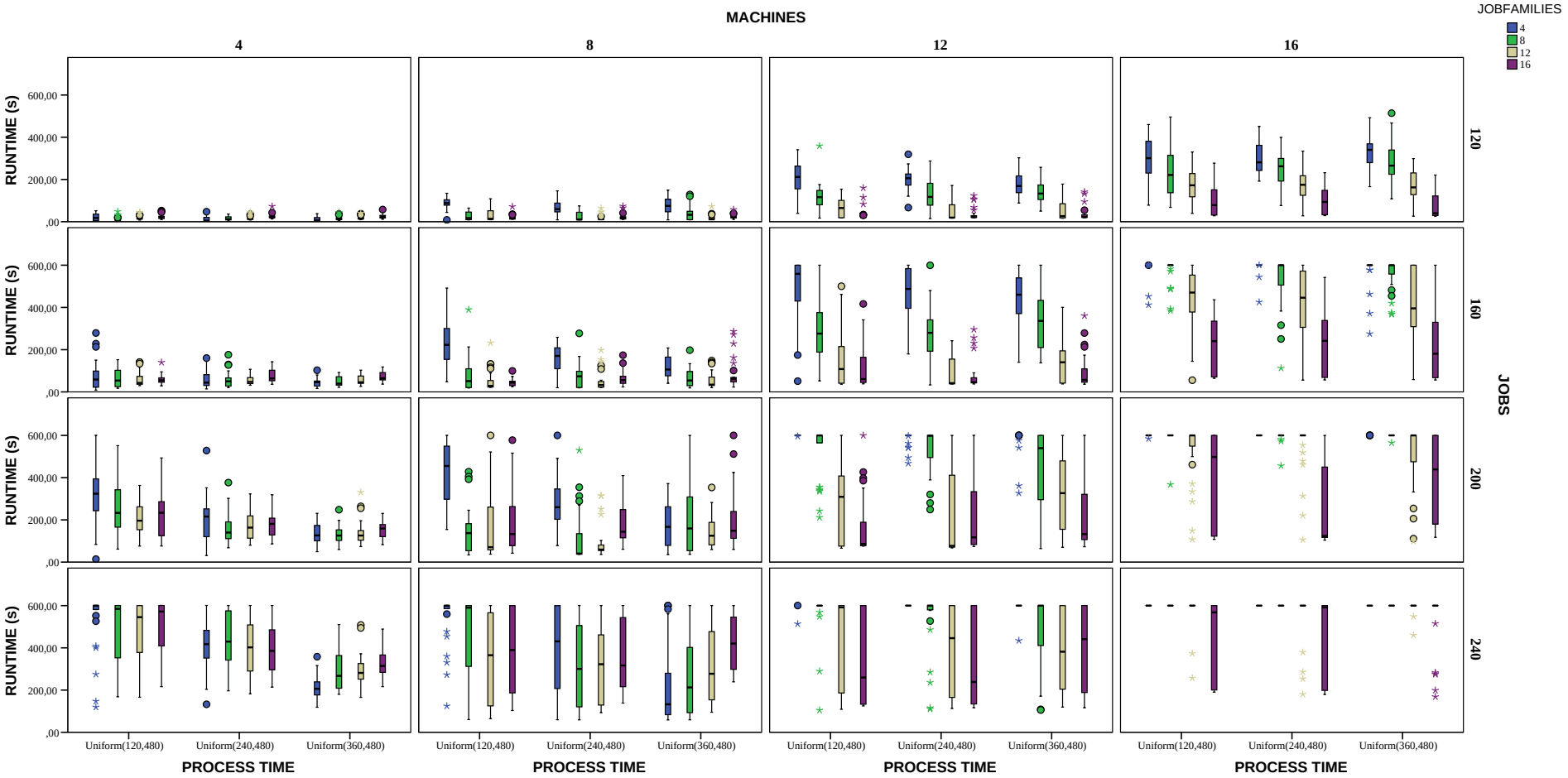
Experiment E2: required run time for cycle time improvements



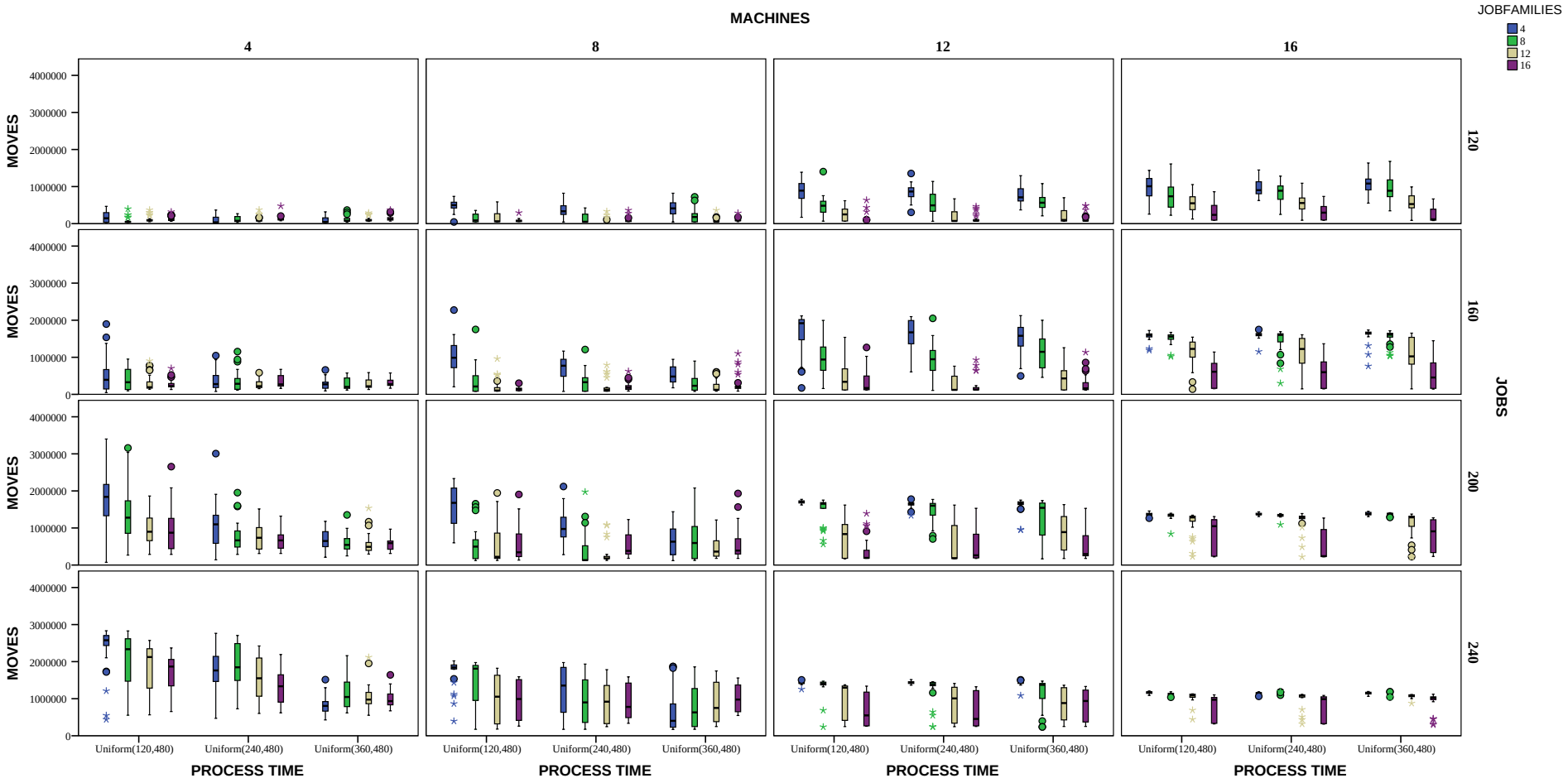
Experiment E2: required moves for cycle time improvements



Experiment E2: tardiness improvements



Experiment E2: required run time for tardiness improvements

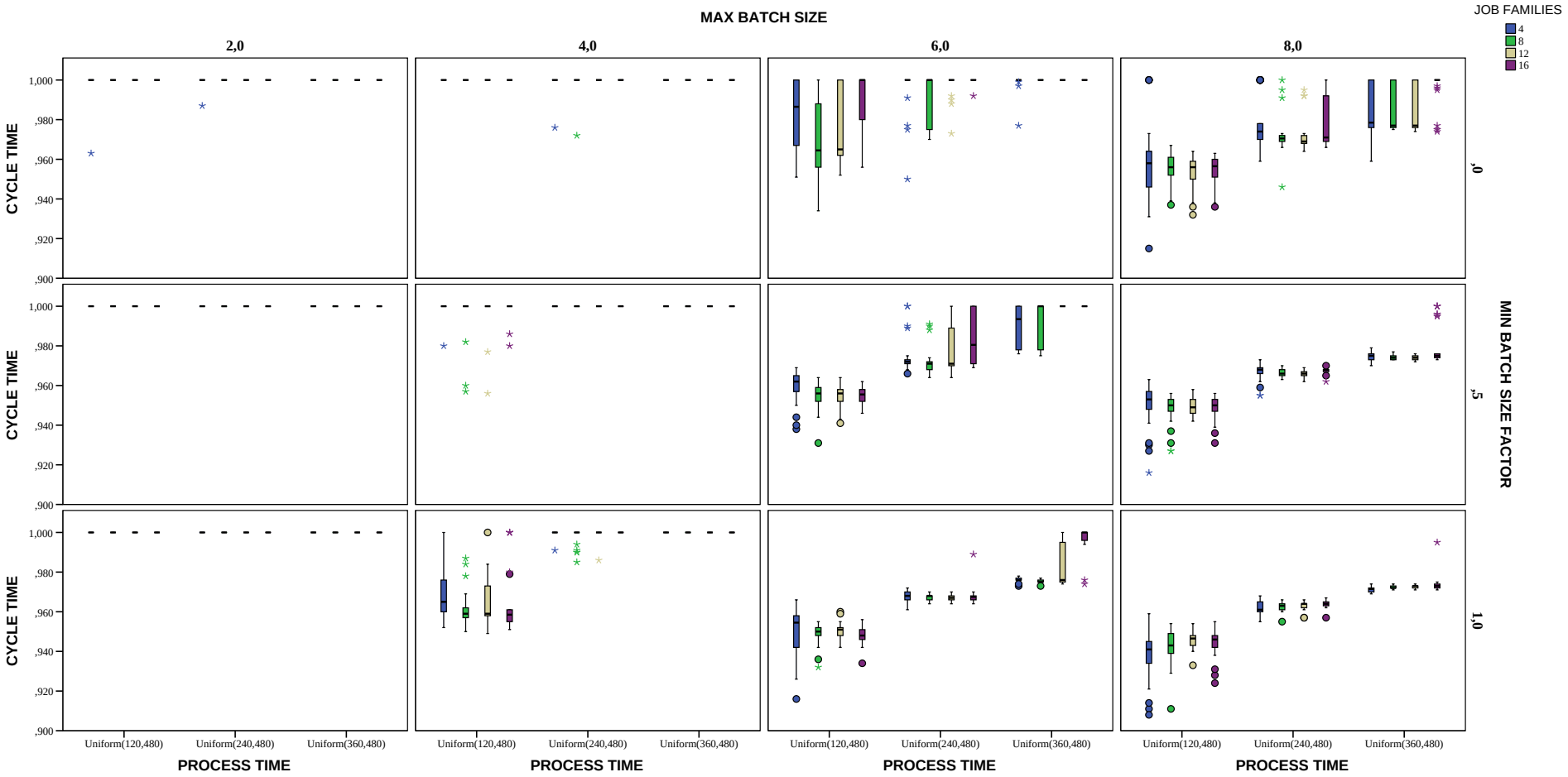


Experiment E2: required moves for tardiness improvements

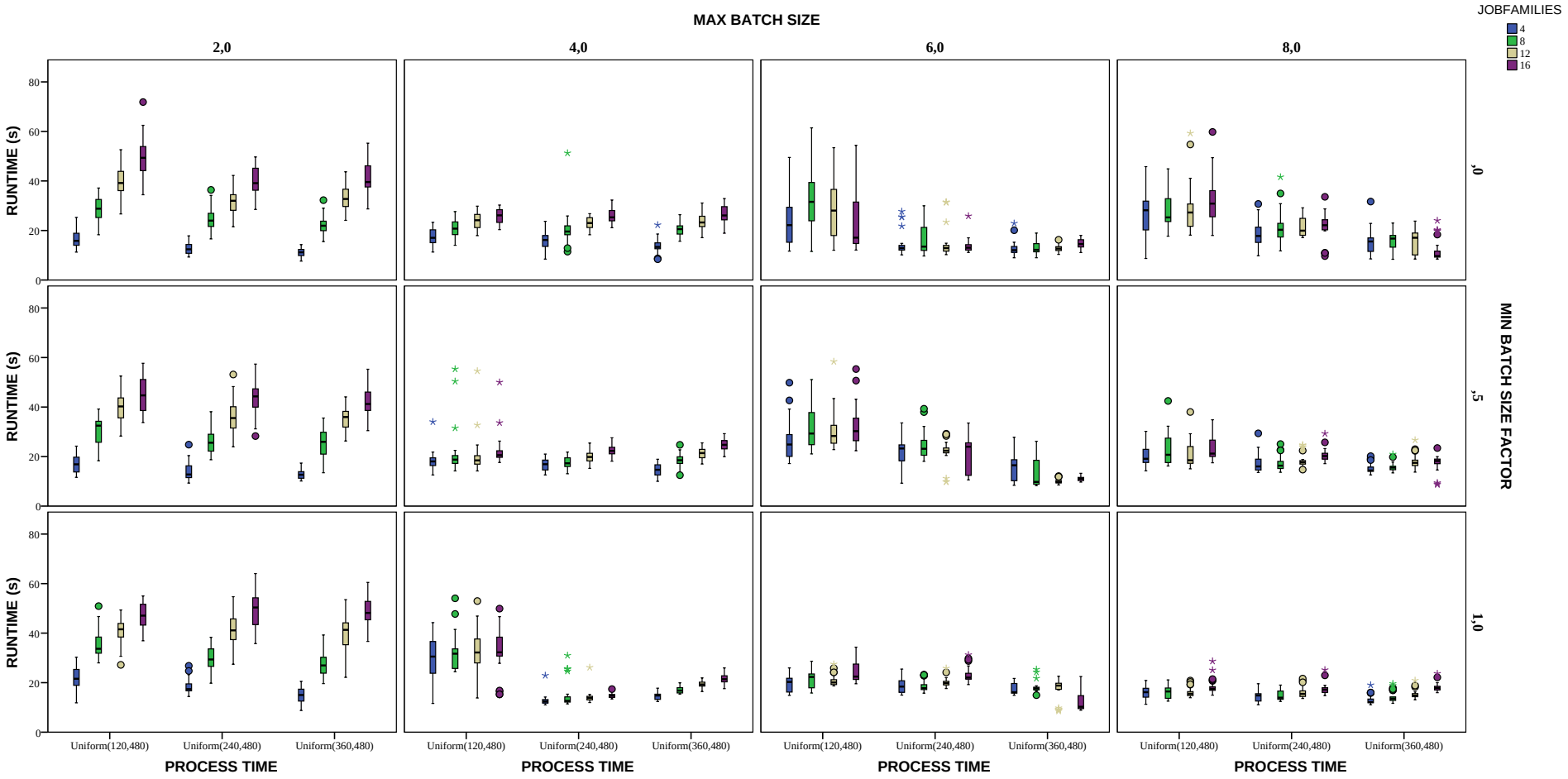
E.3 Experiment E.3

Design of experiment E.3

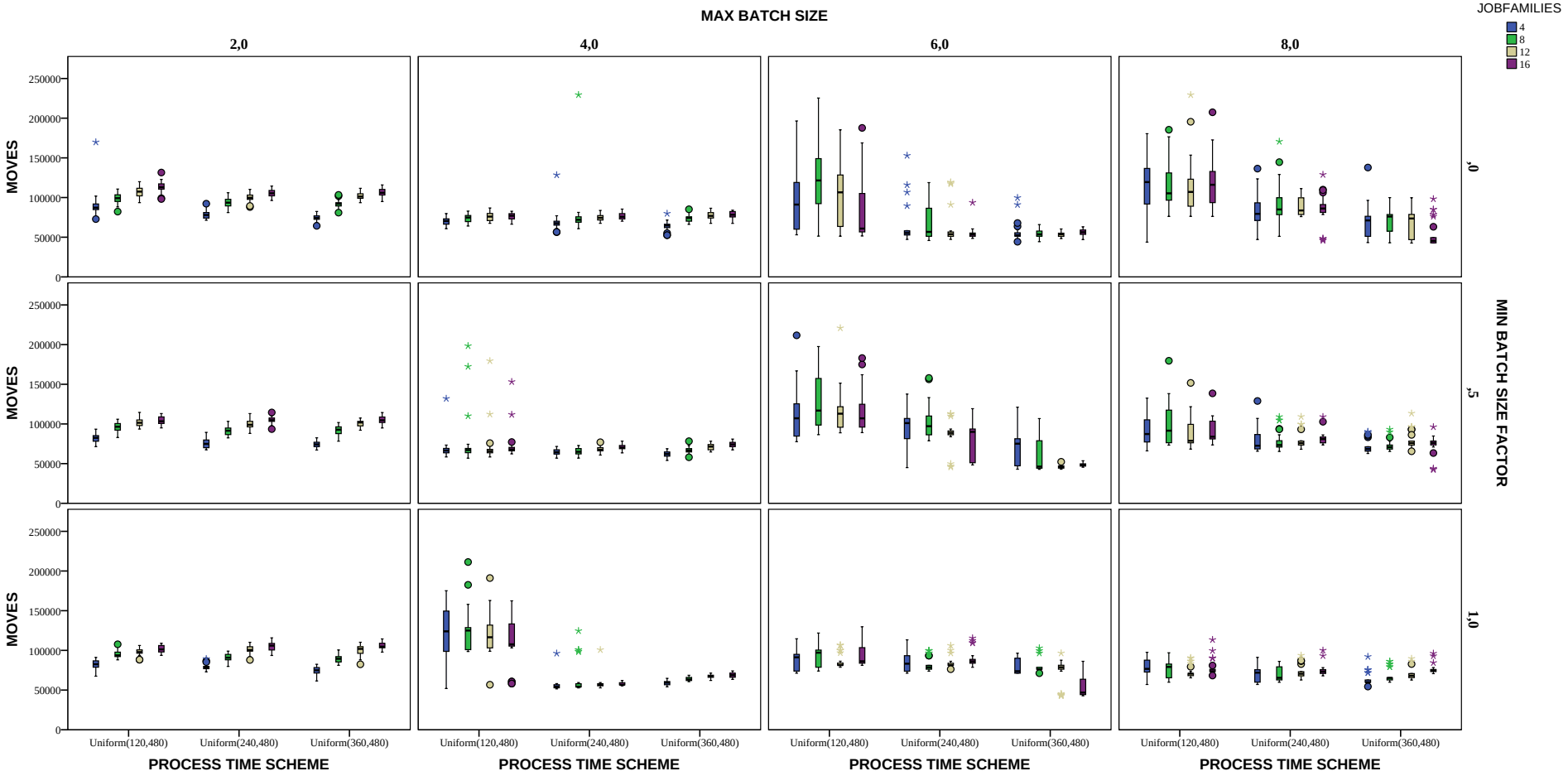
factor	level	count
machines	8	1
jobs	120	1
job families	4, 8, 12, 16	4
dedication density	$1(uniform)$	1
processing times	$U(120, 480), U(240, 480), U(360, 480)$	3
batch size (lots)	(0, 0.5, 1) × (2, 4, 6, 8)	12
batch size (wafers)	unlimited	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	$d_j \in r_j + N(12, 12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		144
model instances per parameter combination		30
total number of model instances		4320
initial solution	EDD (TT) / SPT (TCT)	1
objective(s)	TCT, TT	2
VNS type	VND	1
deadline	10 min	1
number of methods and settings		2
replications per run		1
total number of runs		8640



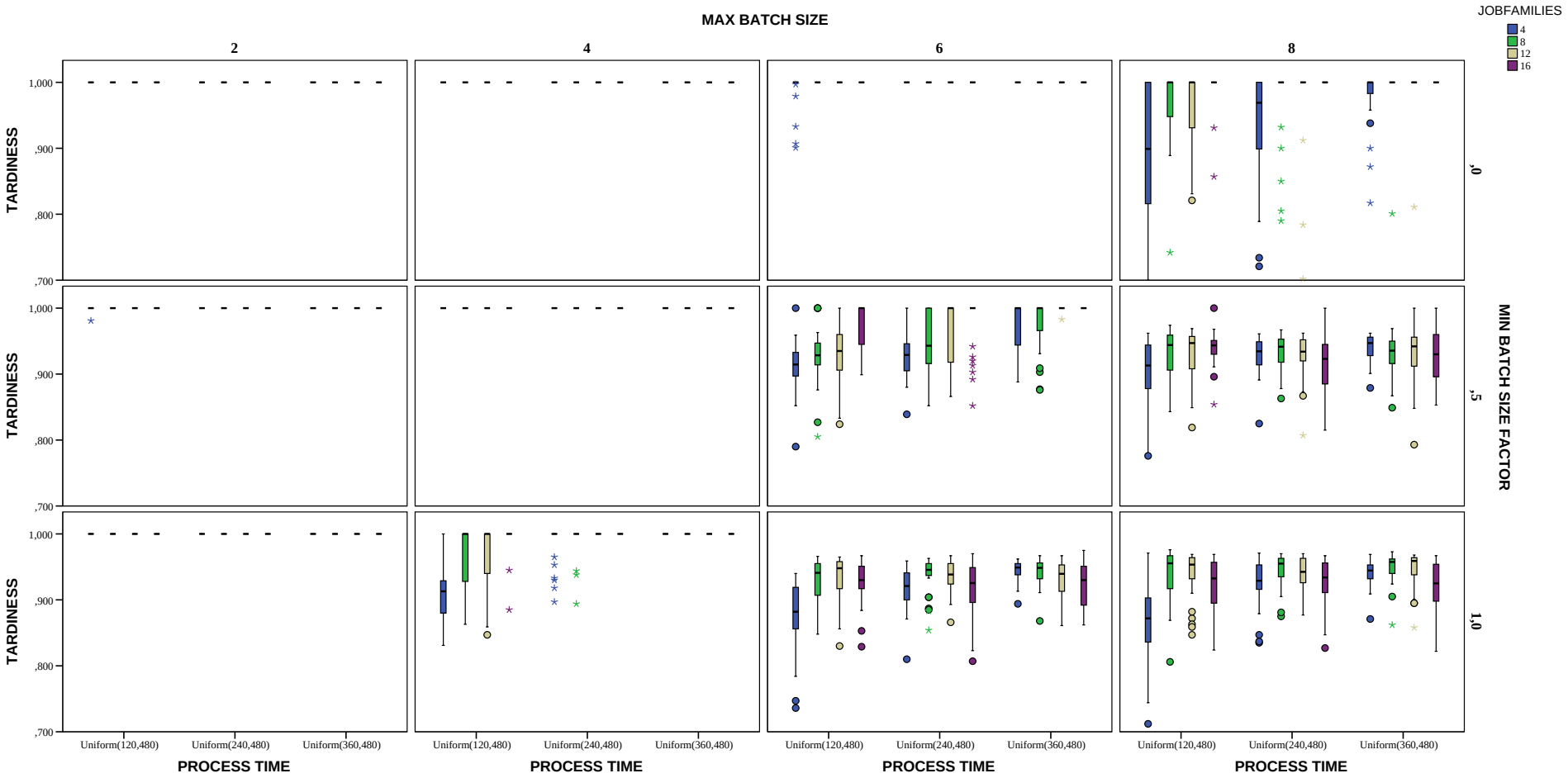
Experiment E3: cycle time improvements



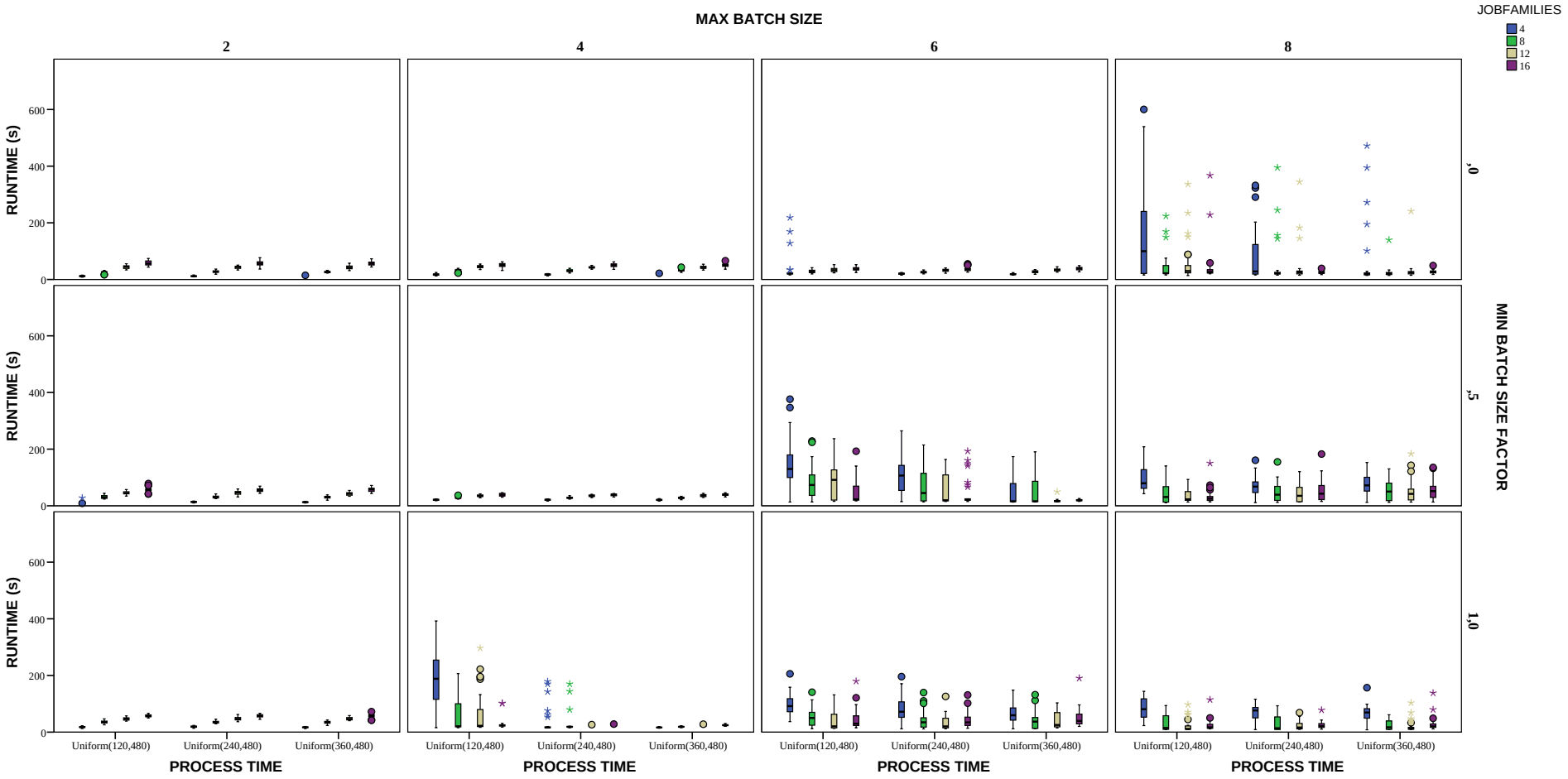
Experiment E3: required run time for cycle time improvements



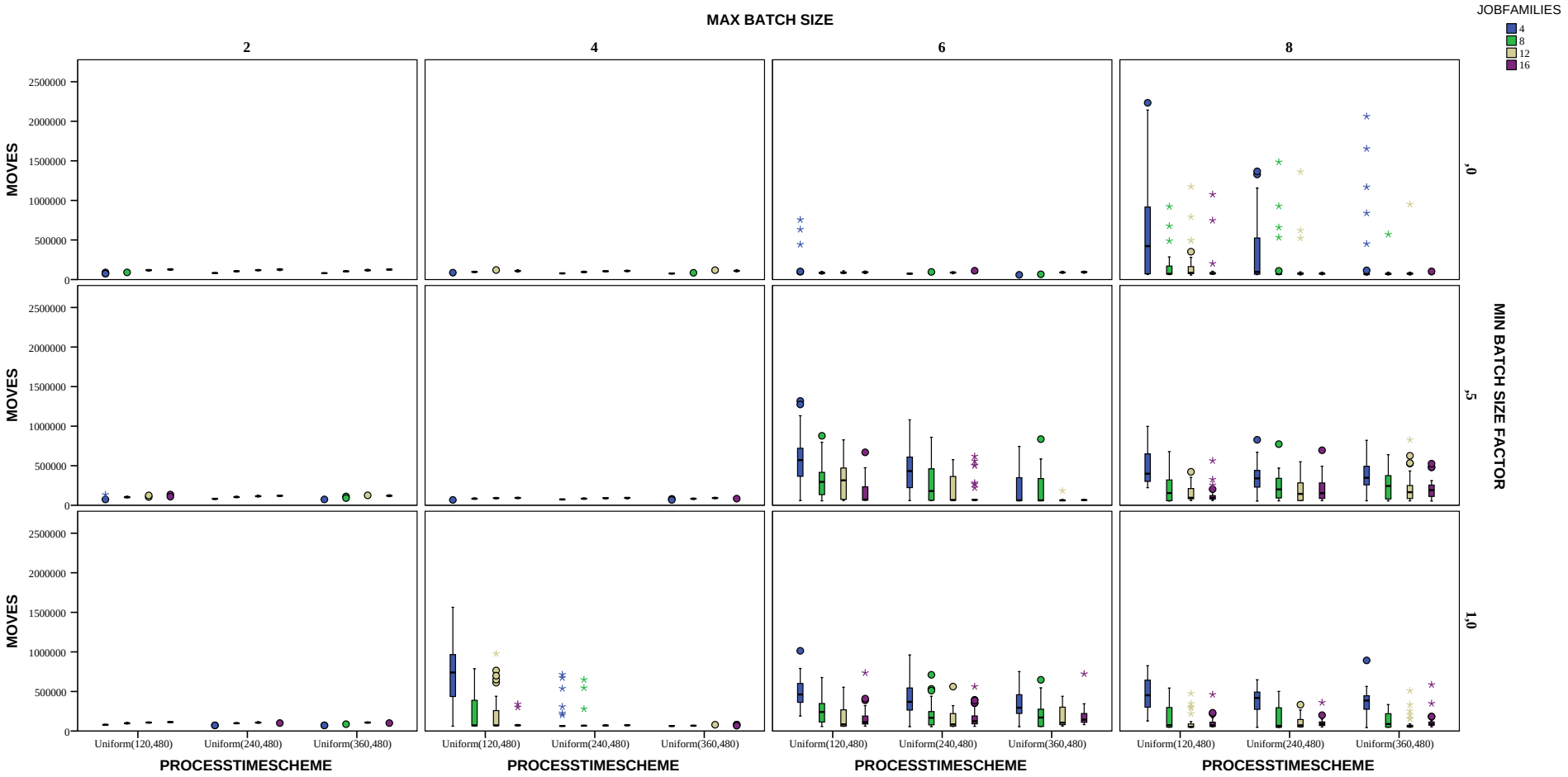
Experiment E3: required moves for cycle time improvements



Experiment E3: tardiness improvements



Experiment E3: required run time for tardiness improvements

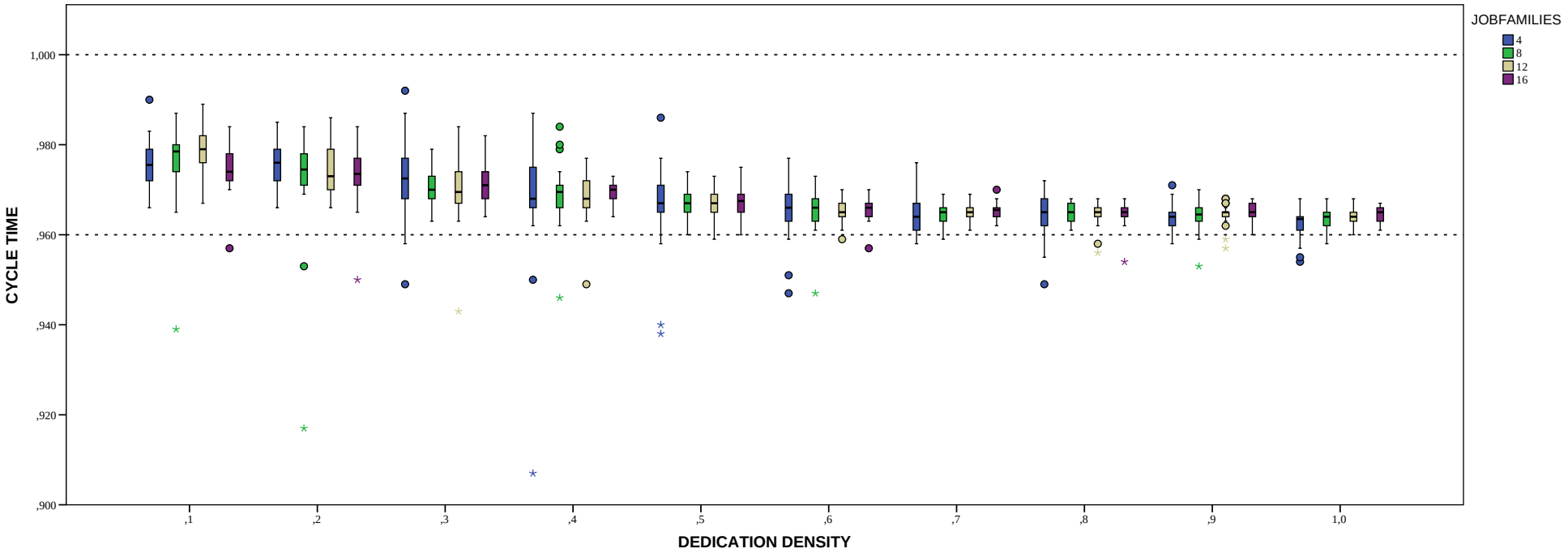


Experiment E3: required moves for tardiness improvements

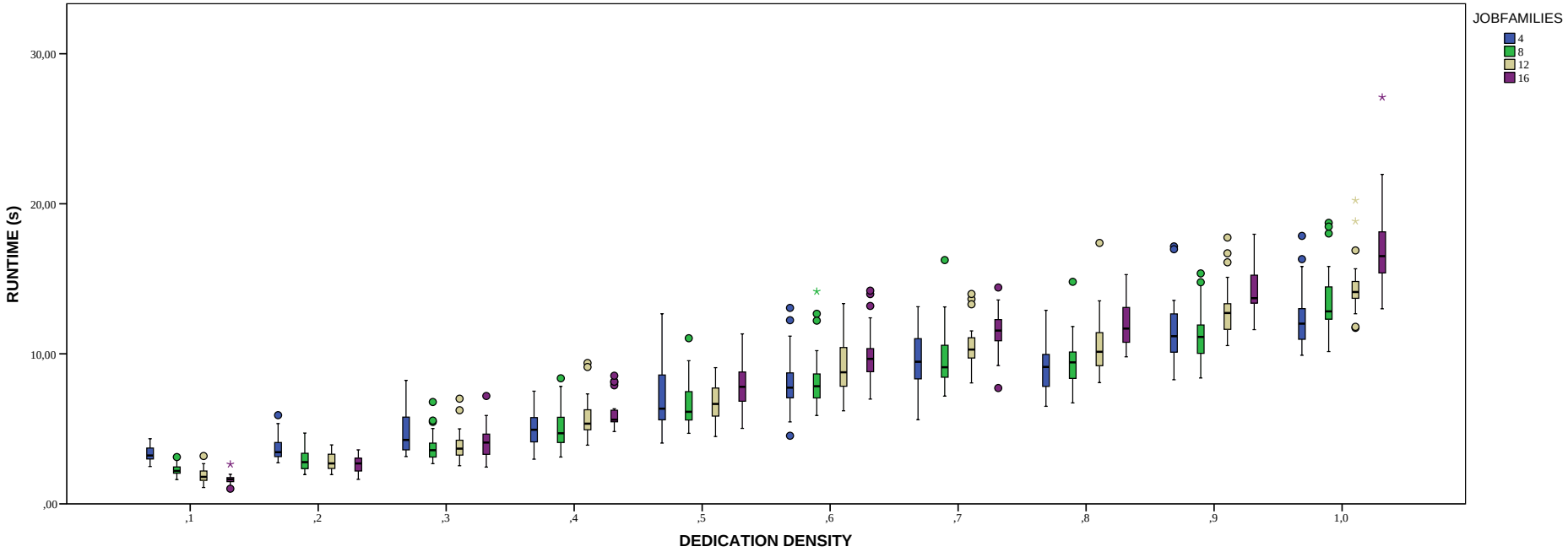
E.4 Experiment E.4

Design of experiment E.4

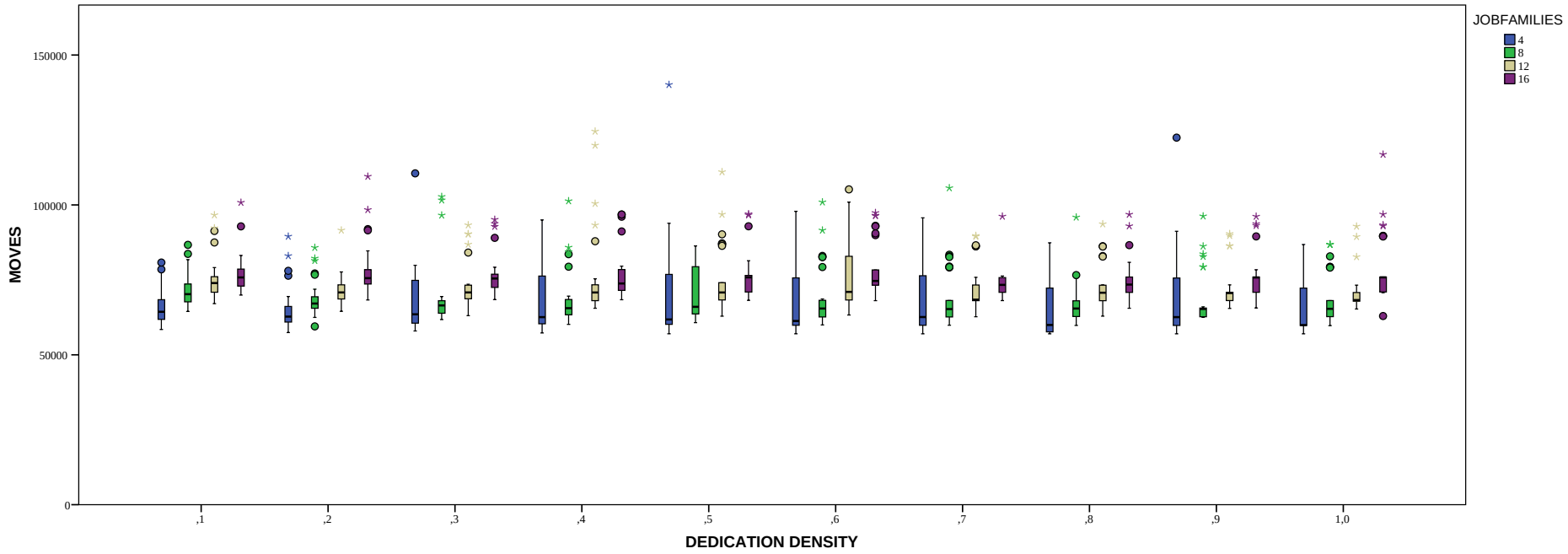
factor	level	count
machines	8	1
jobs	120	1
job families	4, 8, 12, 16	4
dedication density	0.1,0.2, ..., 0.9,1.0	10
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	$d_j \in r_j + N(12, 12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		40
model instances per parameter combination		30
total number of model instances		1200
initial solution	EDD (TT) / SPT (TCT)	1
objective(s)	TCT, TT	2
VNS type	VND	1
deadline	10 min	1
number of methods and settings		2
replications per run		1
total number of runs		2400



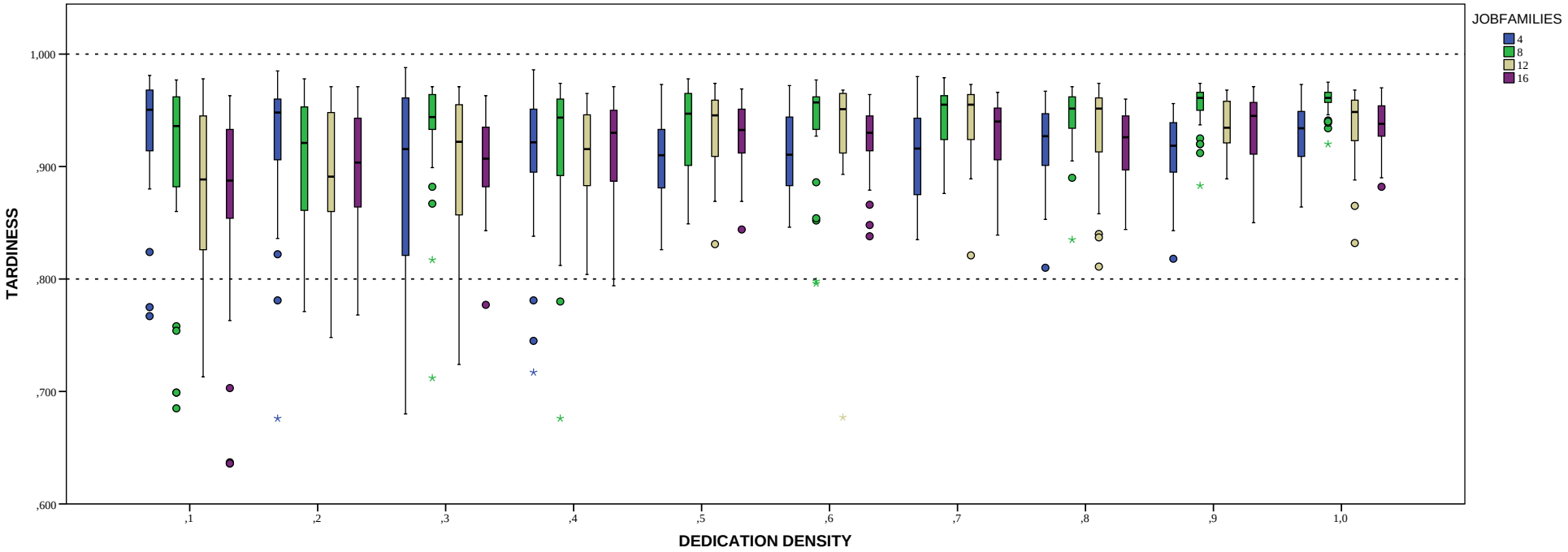
Experiment E4: cycle time improvements



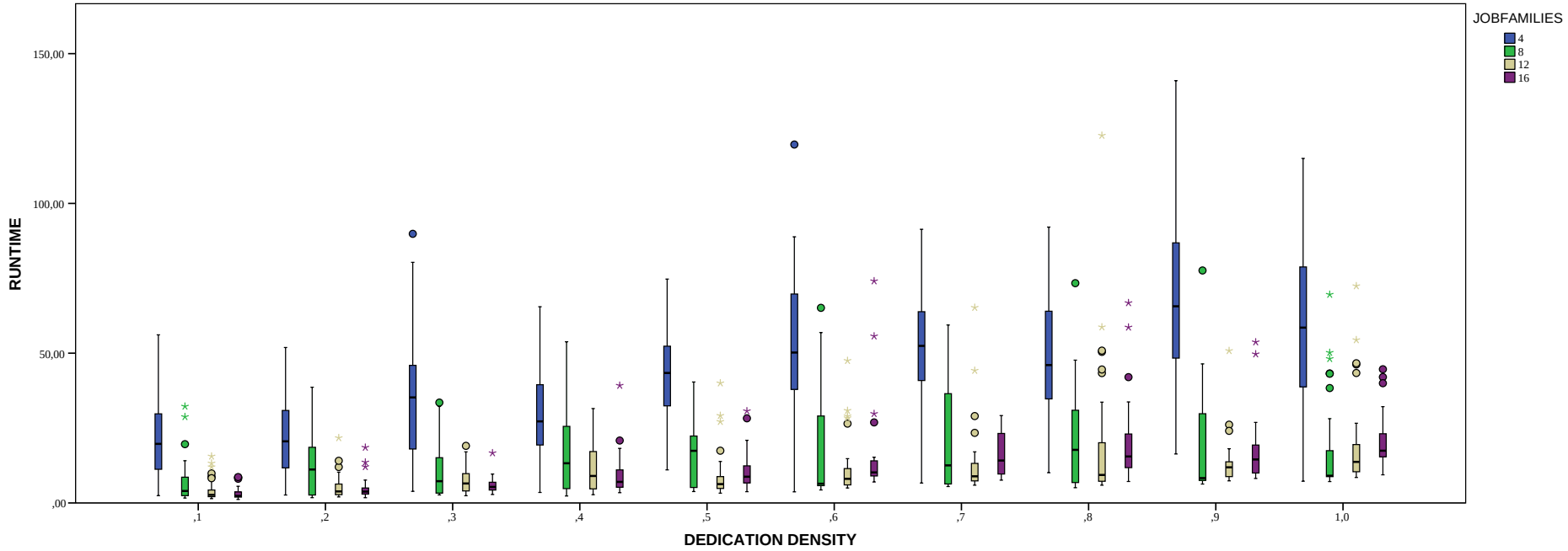
Experiment E4: required run time for cycle time improvements



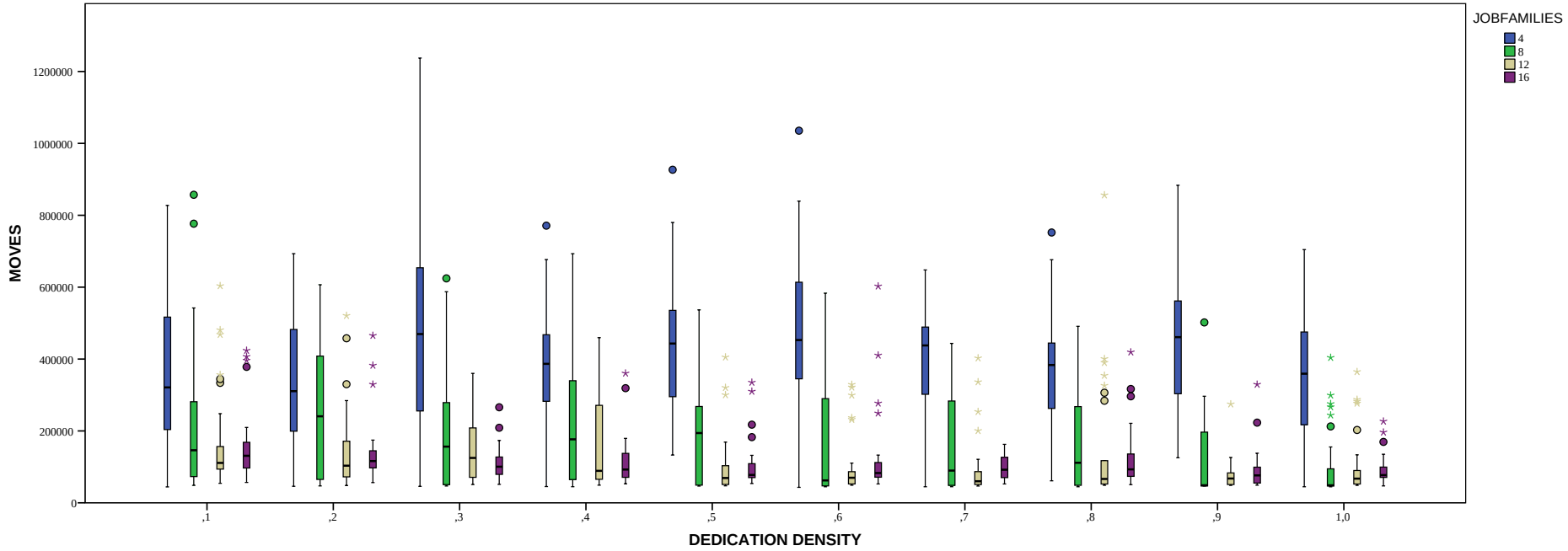
Experiment E4: required moves for cycle time improvements



Experiment E4: tardiness improvements



Experiment E4: required run time for tardiness improvements

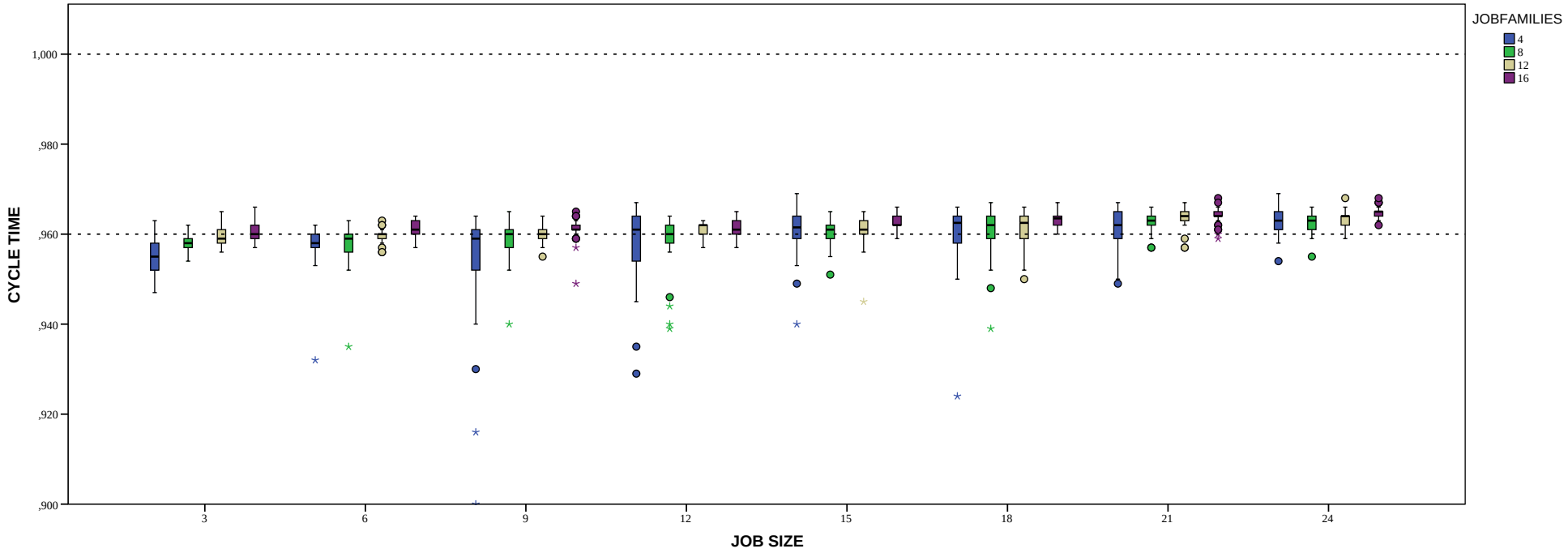


Experiment E4: required moves for tardiness improvements

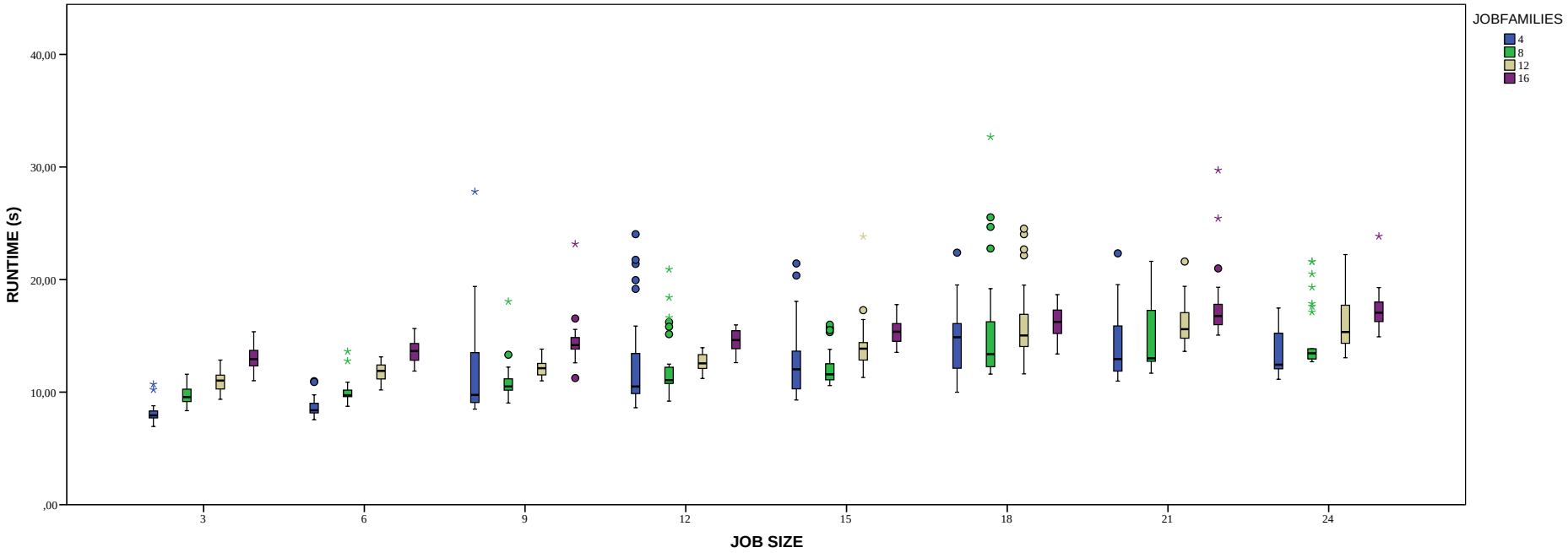
E.5 Experiment E.5

Design of experiment E.5

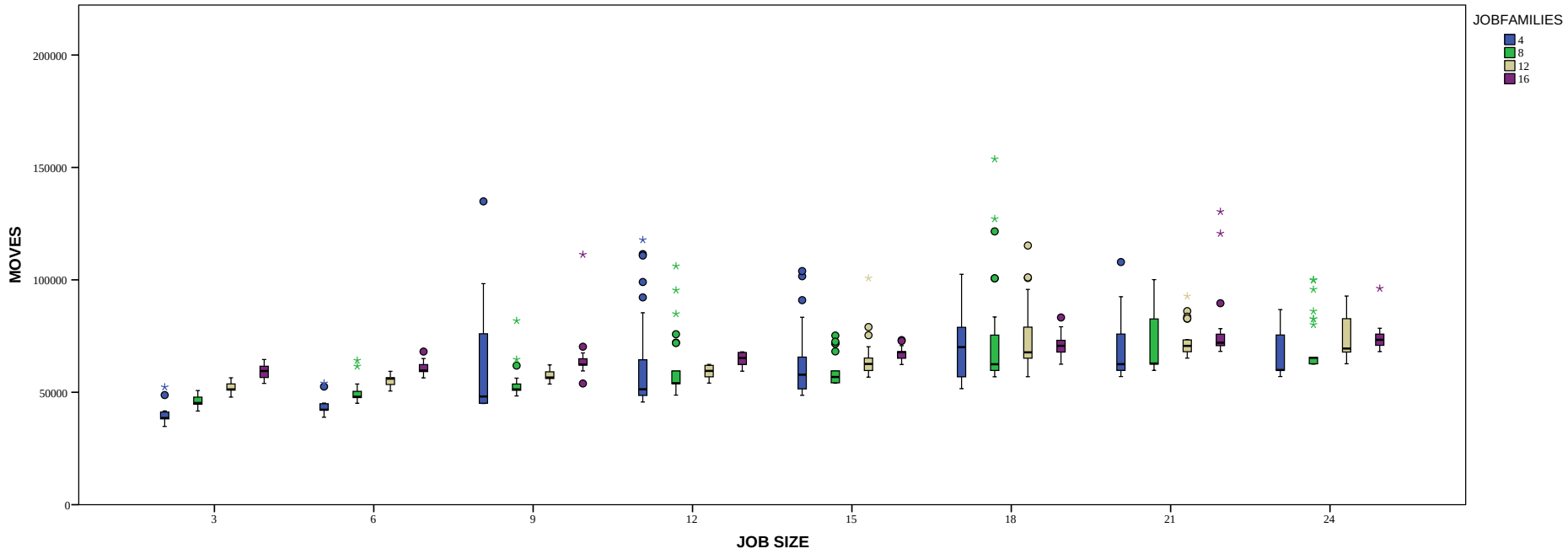
factor	level	count
machines	8	1
jobs	120	1
job families	4, 8, 12, 16	4
dedication density	1 (uniform)	1
processing times	$U(240, 480)$	1
batch size (lots)	unlimited	1
batch size (wafers)	200	1
job sizes	$U(3, 25), U(6, 25), U(9, 25), \ldots, U(24, 25)$	8
job priorities	-	1
job due dates	$d_j \in r_j + N(12, 12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		32
model instances per parameter combination		30
total number of model instances		960
initial solution	EDD (TT) / SPT (TCT)	1
objective(s)	TCT, TT	2
VNS type	VND	1
deadline	10 min	1
number of methods and settings		2
replications per run		1
total number of runs		1920



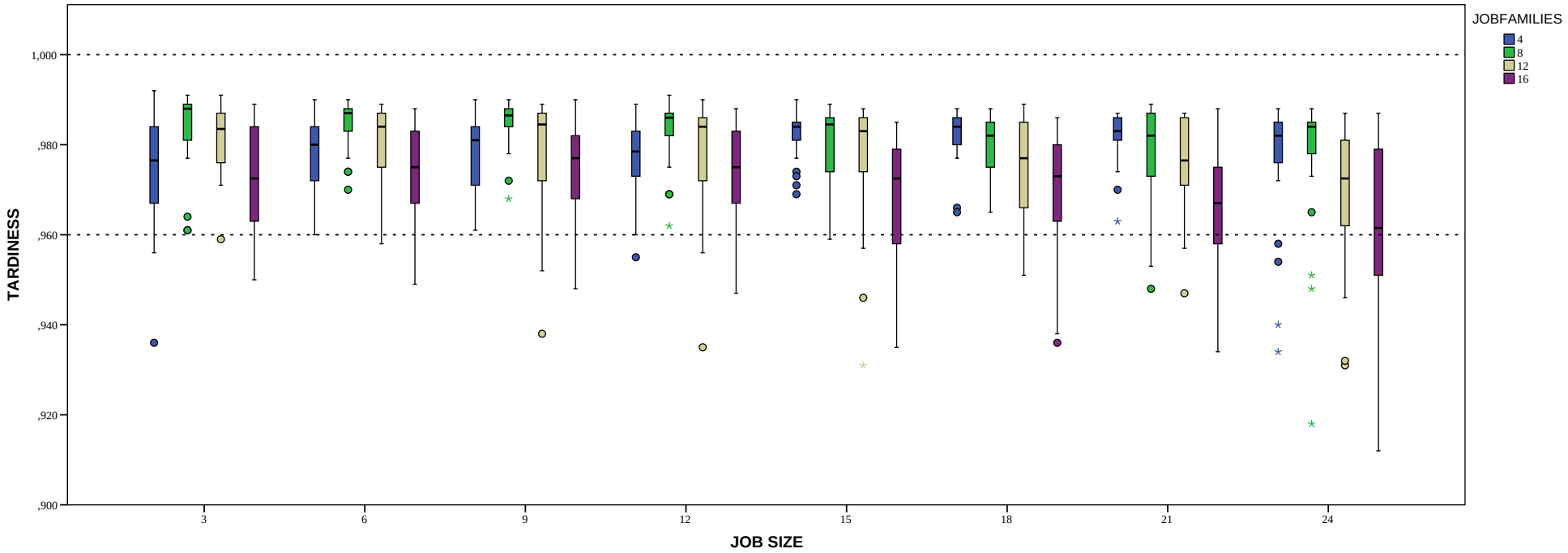
Experiment E5: cycle time improvements



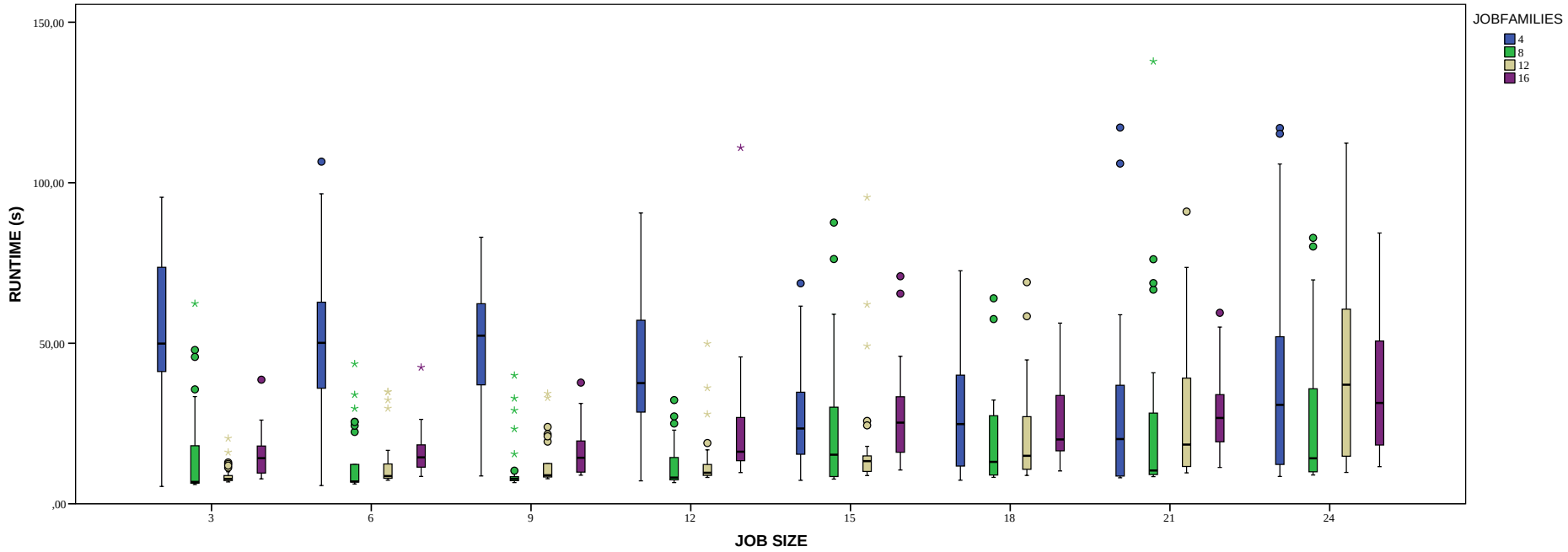
Experiment E5: required run time for cycle time improvements



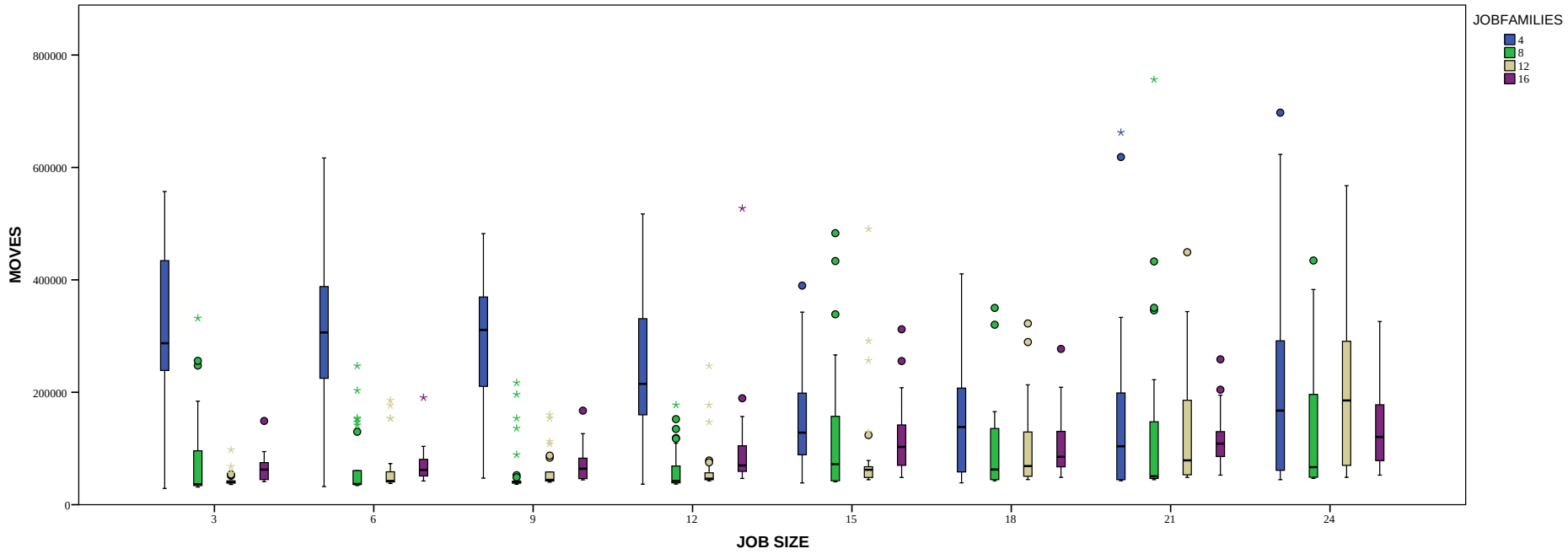
Experiment E5: required moves for cycle time improvements



Experiment E5: tardiness improvements



Experiment E5: required run time for tardiness improvements

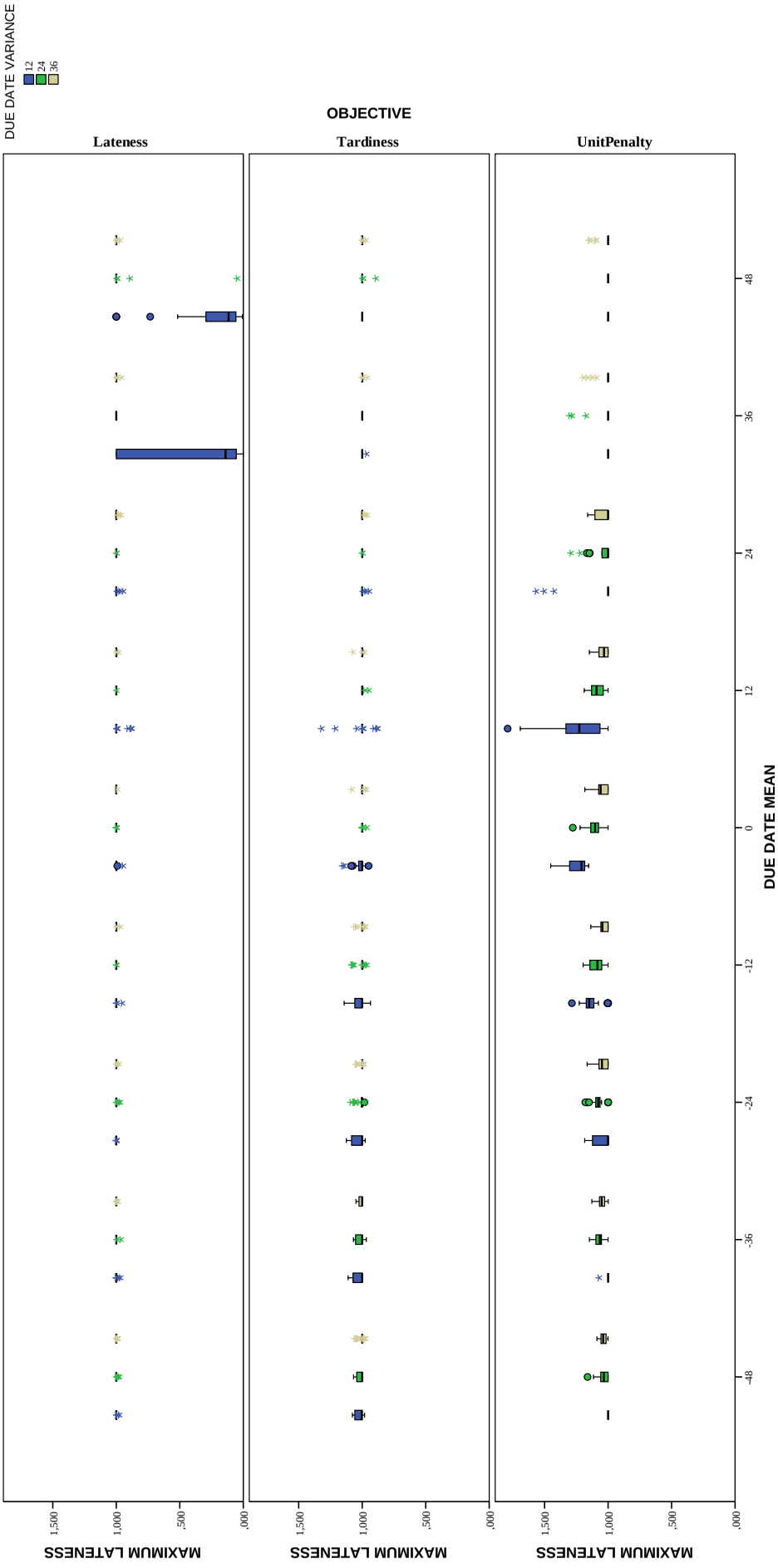


Experiment E5: required moves for tardiness improvements

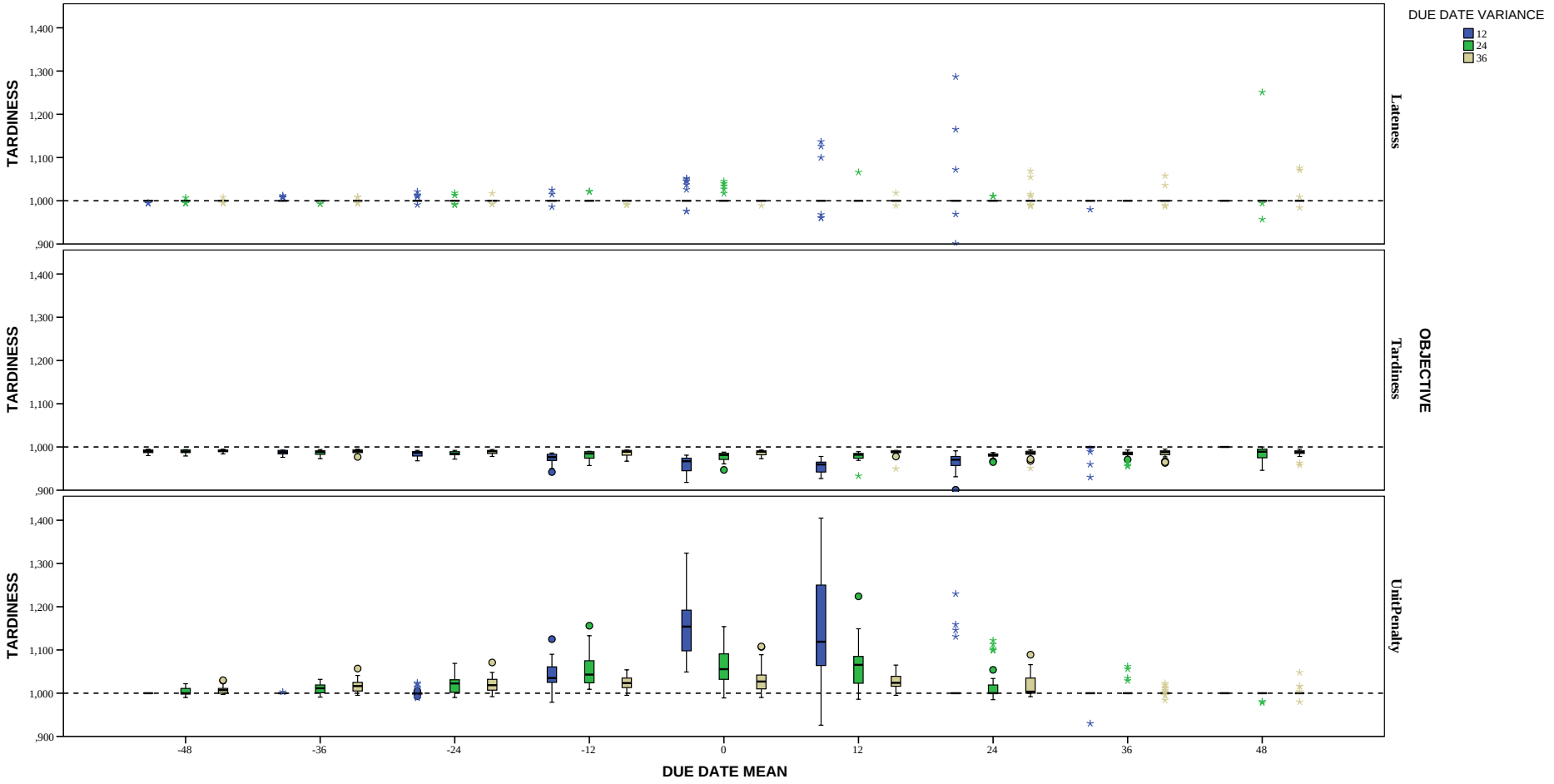
E.6 Experiment E.6

Design of experiment E.6

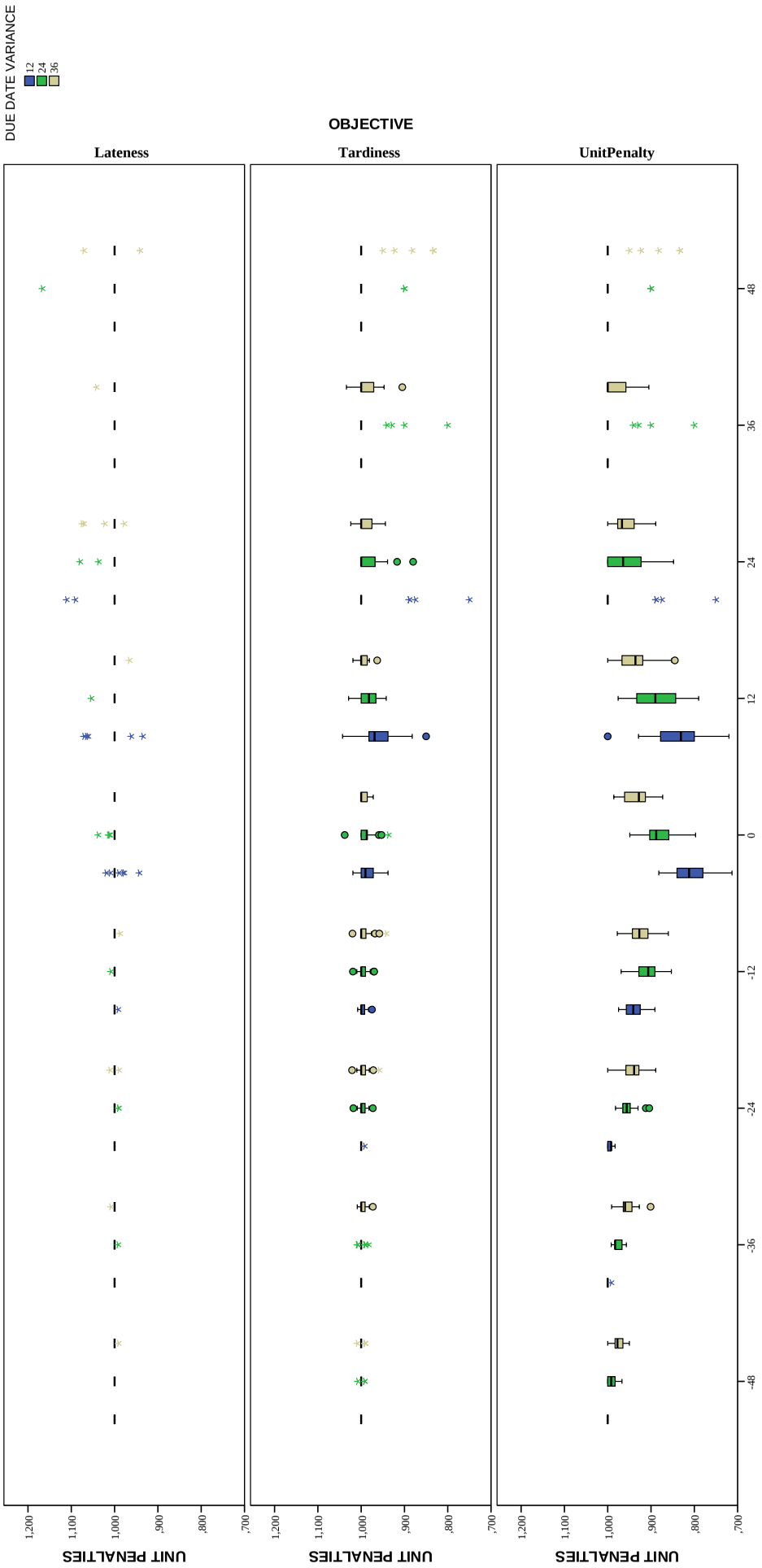
factor	level	count
machines	8	1
jobs	120	1
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	$d_j \in r_j + N(-48, -36, \ldots, 48; 12, 24, 36)$	27
job arrivals	$r_j = 0$	1
problem parameter combinations		27
model instances per parameter combination		30
total number of model instances		810
initial solution	EDD	1
objective(s)	L_{max}, TT, TU	3
VNS type	VND	1
deadline	10 min	1
number of methods and settings		3
replications per run		1
total number of runs		2430



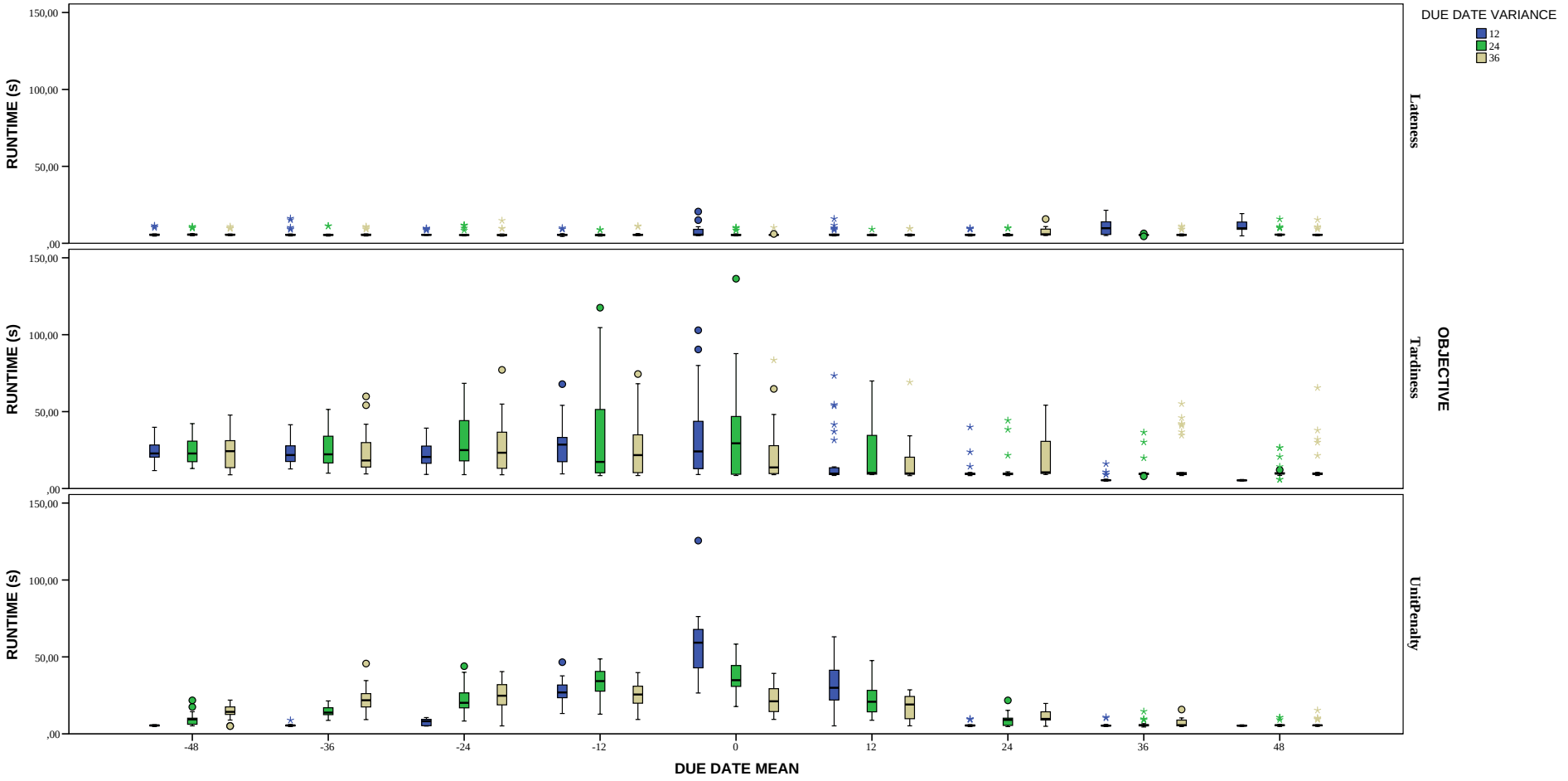
Experiment E6: lateness improvements



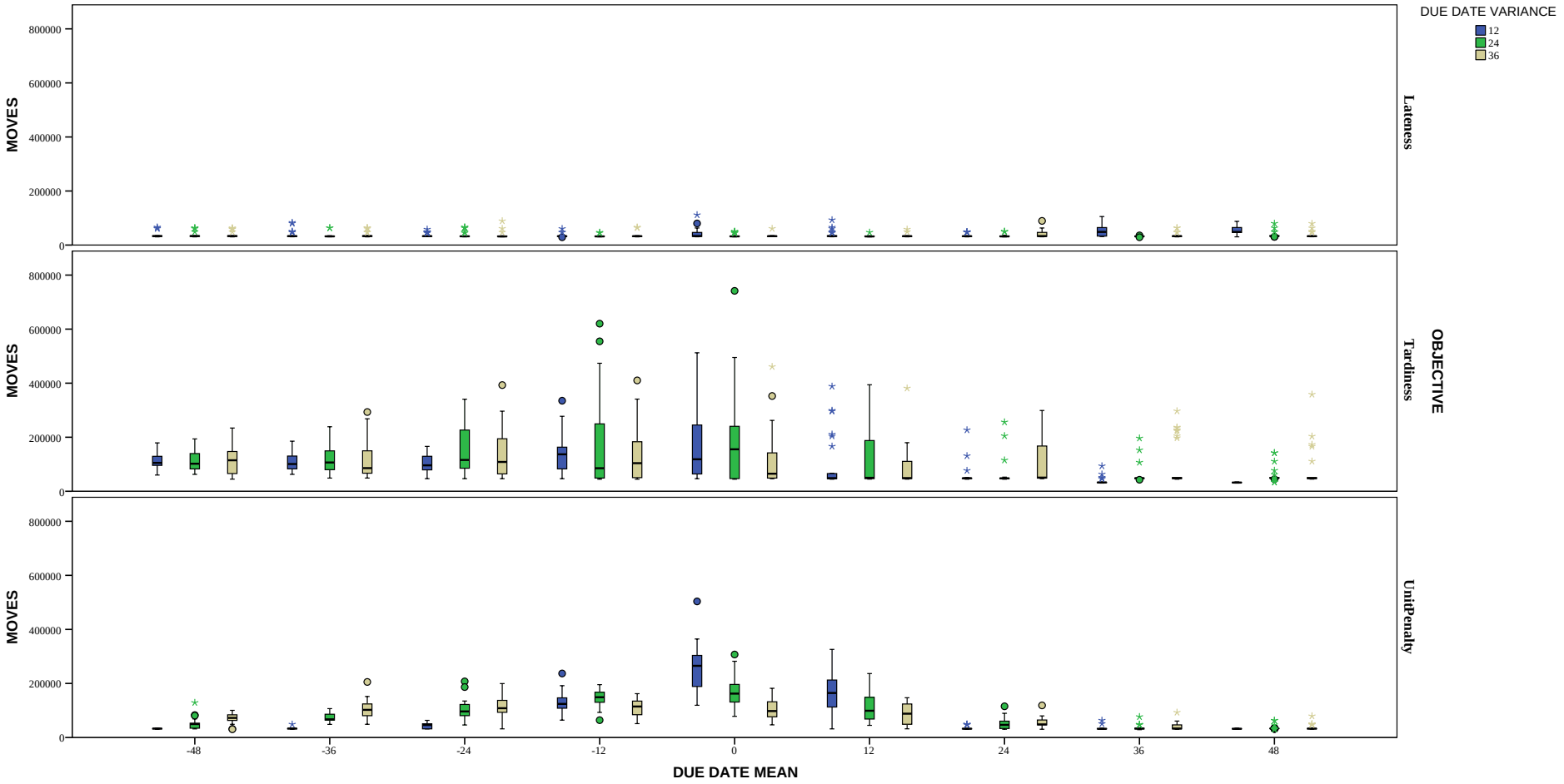
Experiment E6: tardiness improvements



Experiment E6: unit penalty improvements



Experiment E6: run times for improving objectives related to due dates

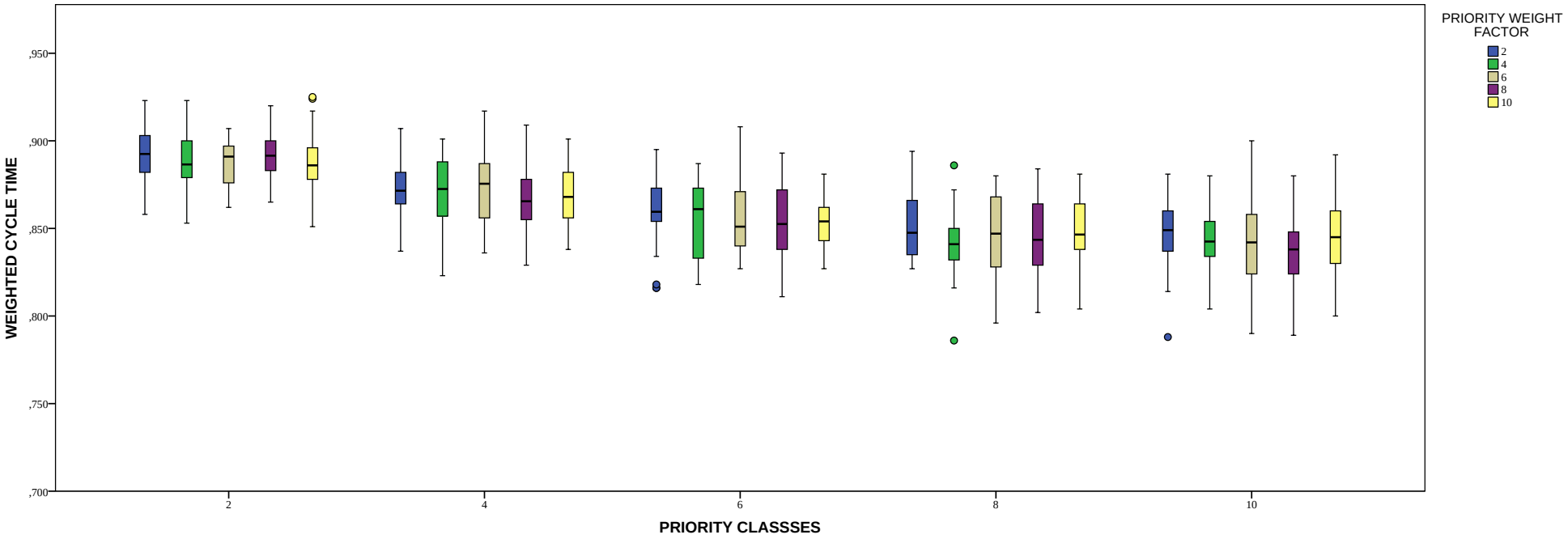


Experiment E6: moves for improving objectives related to due dates

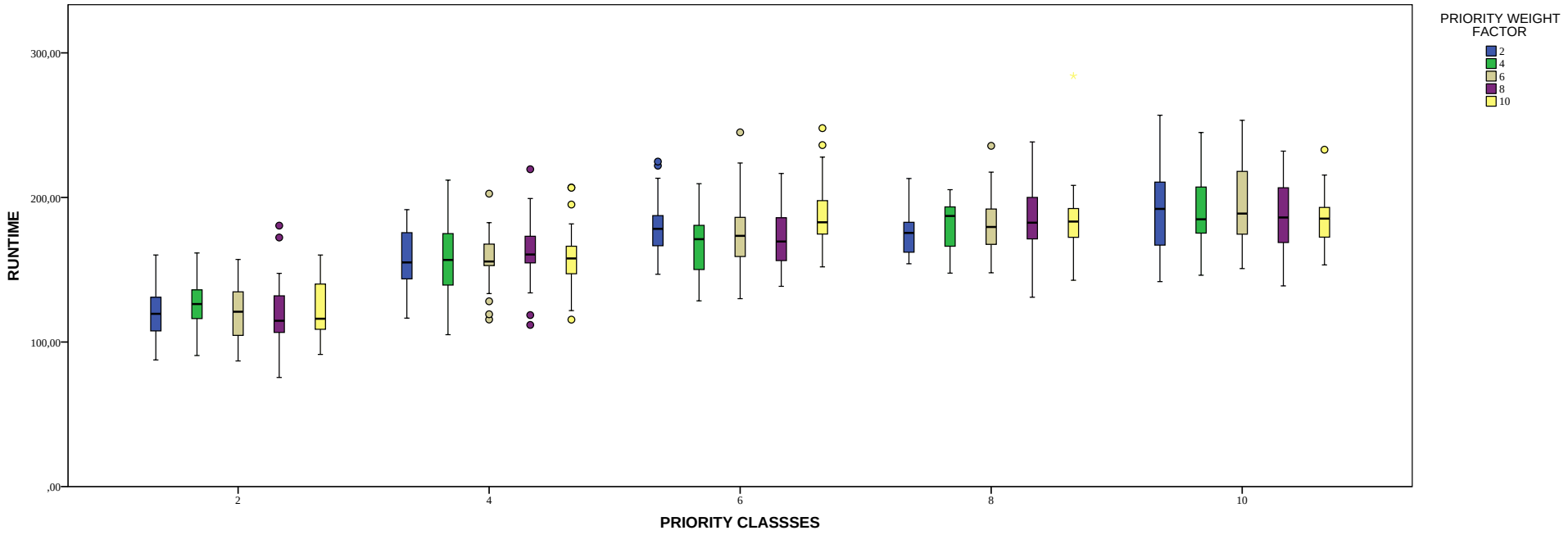
E.7 Experiment E.7

Design of experiment E.7

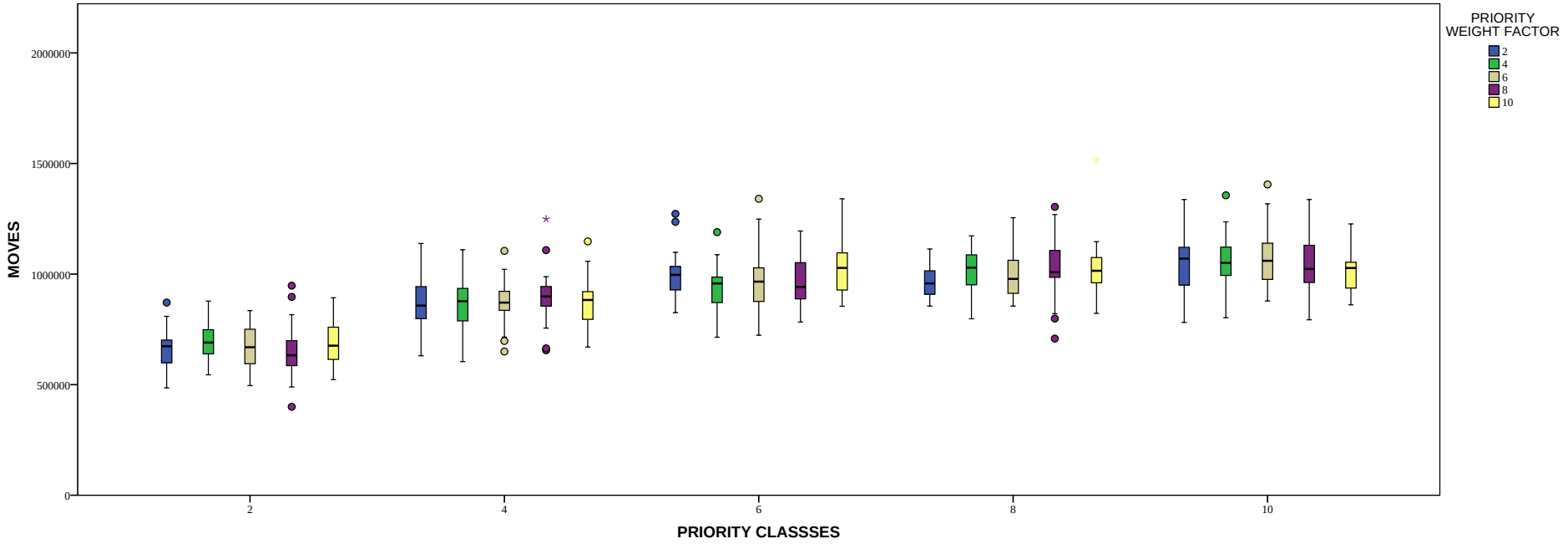
factor	level	count
machines	8	1
jobs	120	1
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities (classes)	2, 4, 6, 8, 10	5
job priorities (weights)	$w_j \in U(1; 2, 4, 6, 8, 10)$	5
job due dates	$d_j \in r_j + N(12, 12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		25
model instances per parameter combination		30
total number of model instances		750
initial solution	WEDD (TT) / WSPT (TCT)	1
objective(s)	TWCT, TWT	2
VNS type	VND	1
deadline	10 min	1
number of methods and settings		2
replications per run		1
total number of runs		1500



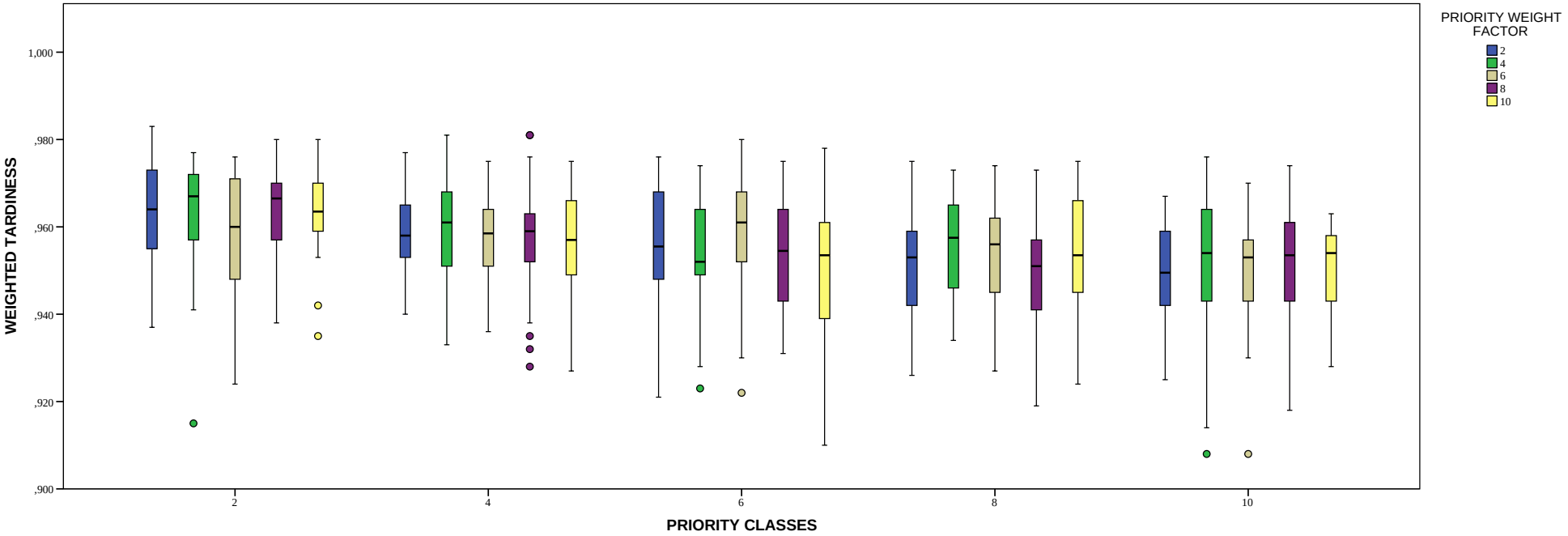
Experiment E7: weighted cycle time improvements



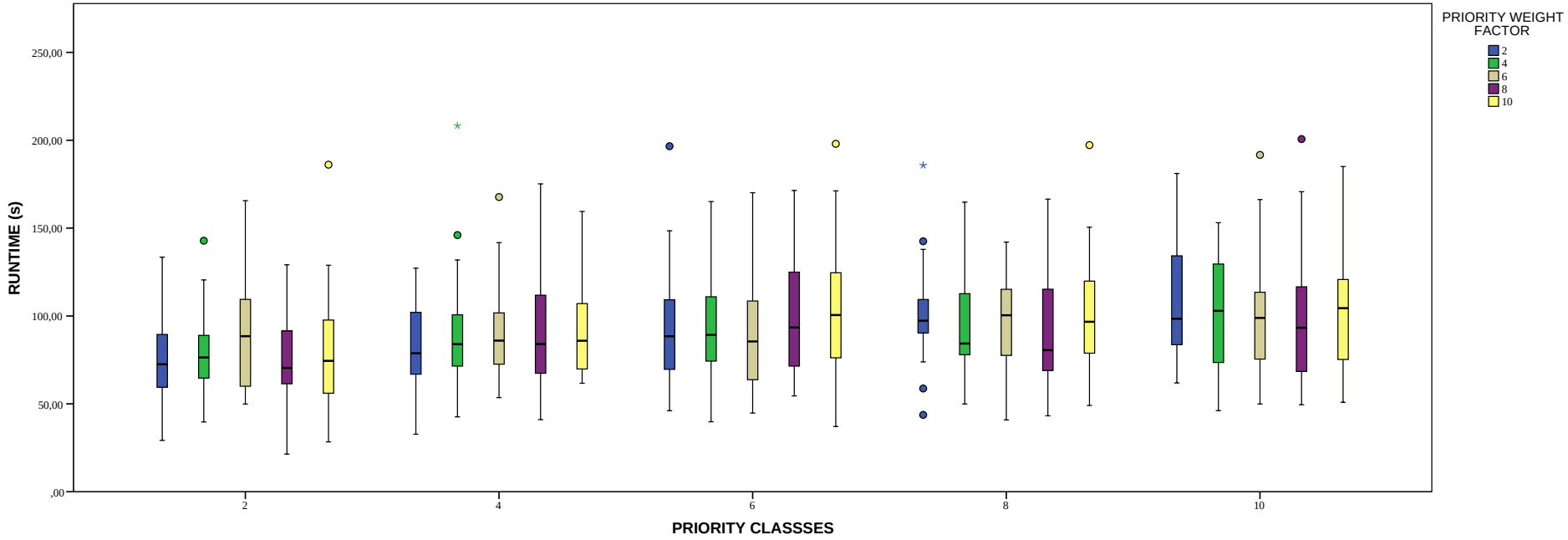
Experiment E7: required run time for weighted cycle time improvements



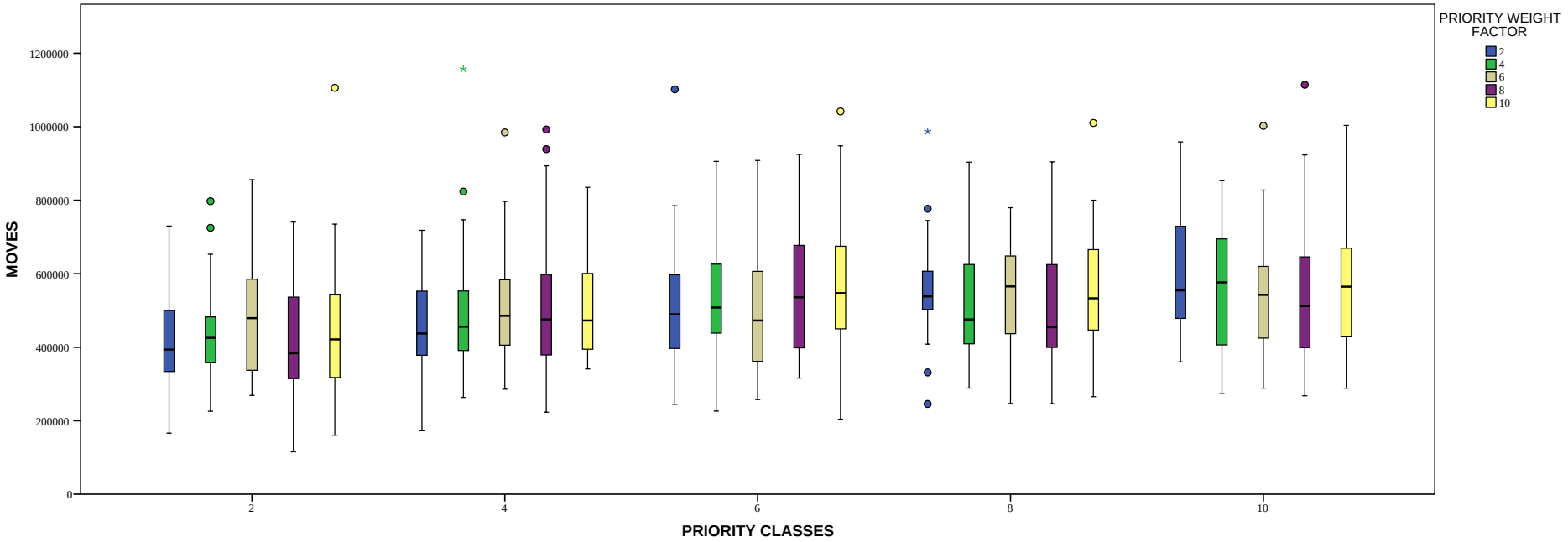
Experiment E7: required moves for weighted cycle time improvements



Experiment E7: weighted tardiness improvements



Experiment E7: required run time for weighted tardiness improvements

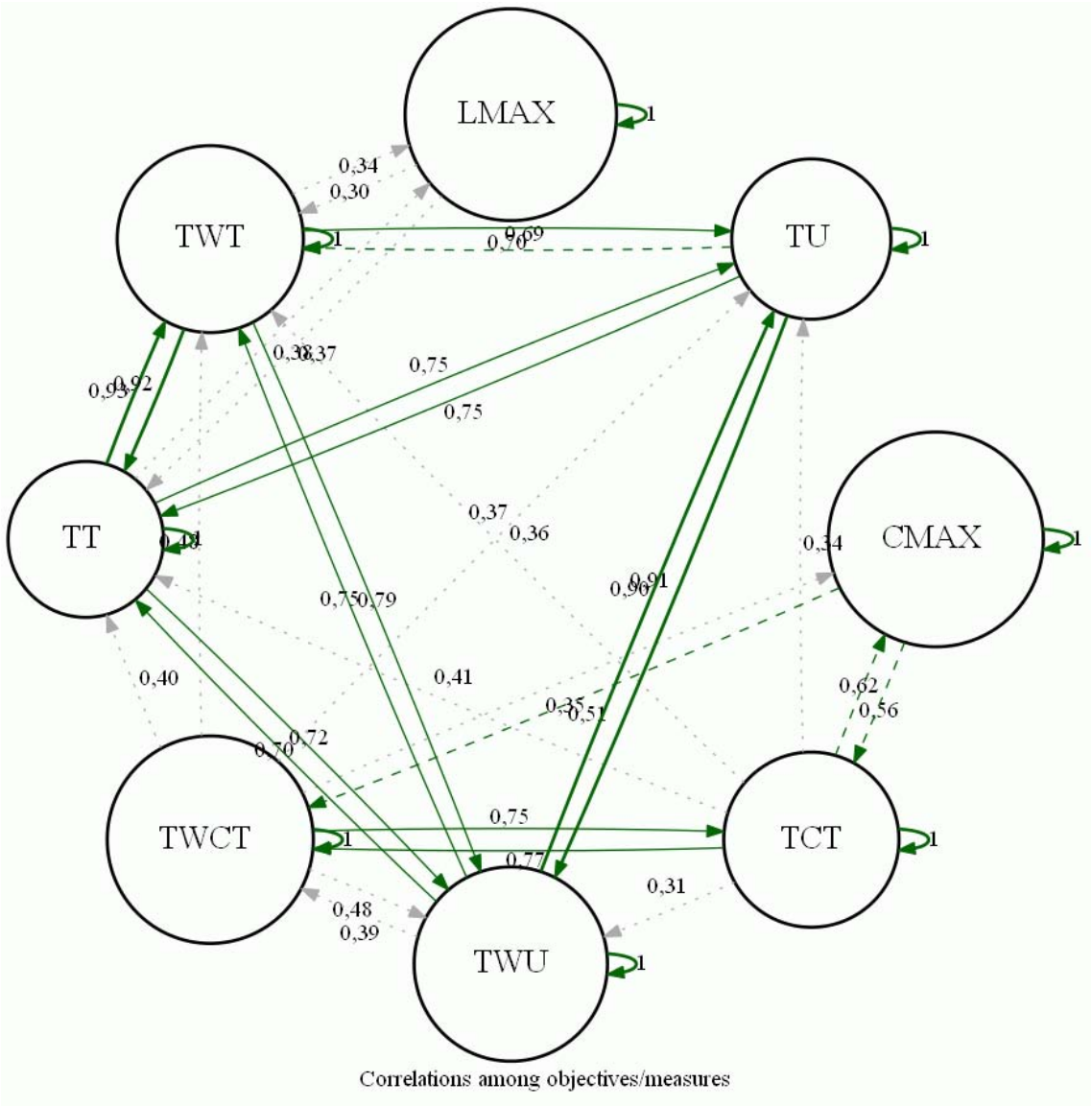


Experiment E7: required moves for weighted tardiness improvements

E.8 Experiment E.8

Design of experiment E.8

factor	level	count
machines	8	1
jobs	120	1
job families	4, 8, 12, 16	4
dedication density	$1(uniform)$	1
processing times	$U(120,480), U(240,480), U(360,480)$	3
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	$w_j \in U(1;5)$	1
job due dates	$d_j \in r_j + N(12,12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		12
model instances per parameter combination		30
total number of model instances		360
initial solution	random	1
objective(s)	C_{max}, L_{max}, TCT, TU, TT, TWCT, TWU, TWT	8
VNS type	VND	1
deadline	10 min	1
number of methods and settings		8
replications per run		1
total number of runs		2880

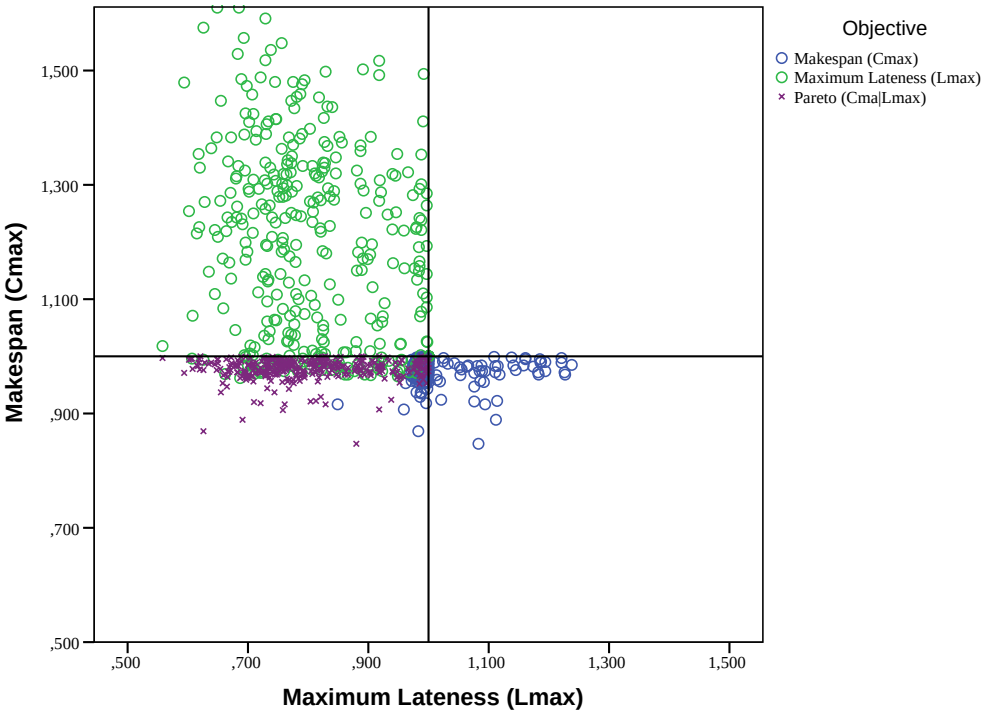


Experiment E8: correlation factors between objectives

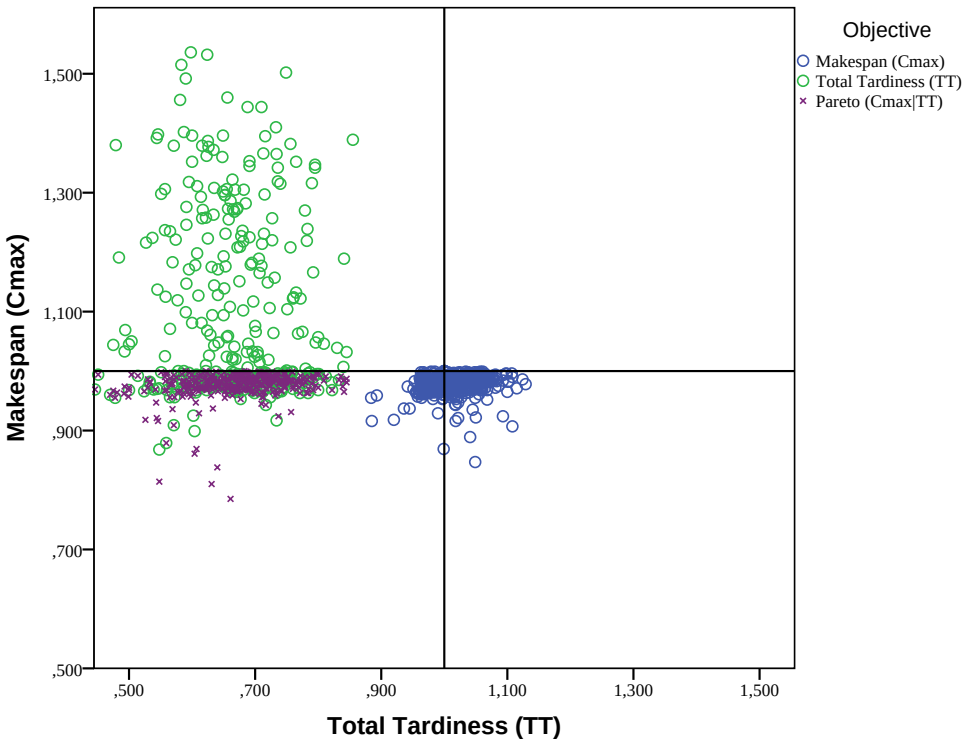
E.9 Experiment E.9

Design of experiment E.9

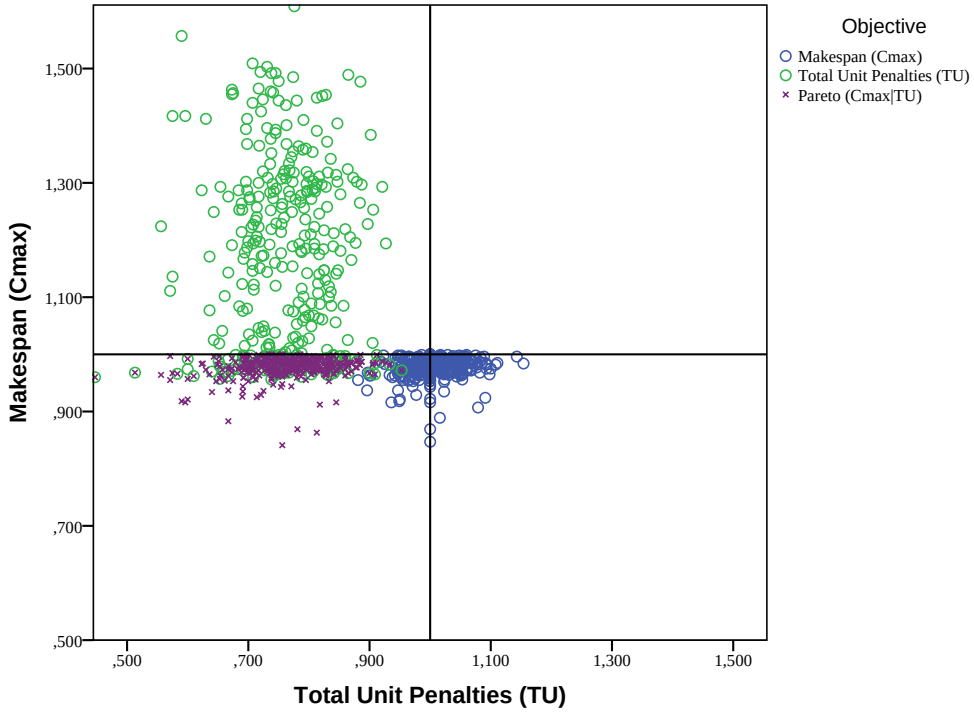
factor	level	count
machines	8	1
jobs	120	1
job families	4, 8, 12, 16	4
dedication density	$1(uniform)$	1
processing times	$U(120,480), U(240,480), U(360,480)$	3
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	$w_j \in U(1;5)$	1
job due dates	$d_j \in r_j + N(12,12)$	1
job arrivals	$r_j = 0$	1
problem parameter combinations		12
model instances per parameter combination		30
total number of model instances		360
initial solution	random	1
objective(s)	C_{max}, L_{max}, TCT, TU, TT, TWCT, TWU, TWT	15
VNS type	VND	1
deadline	10 min	1
number of methods and settings		15
replications per run		1
total number of runs		5400



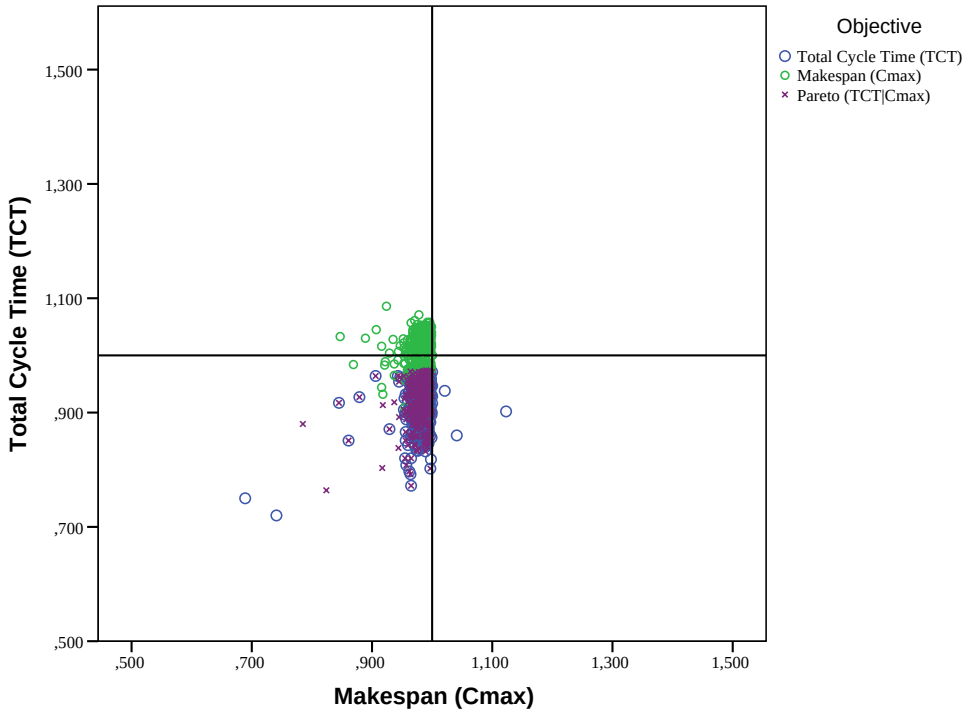
Experiment E9: pareto optimization of makespan and maximum lateness



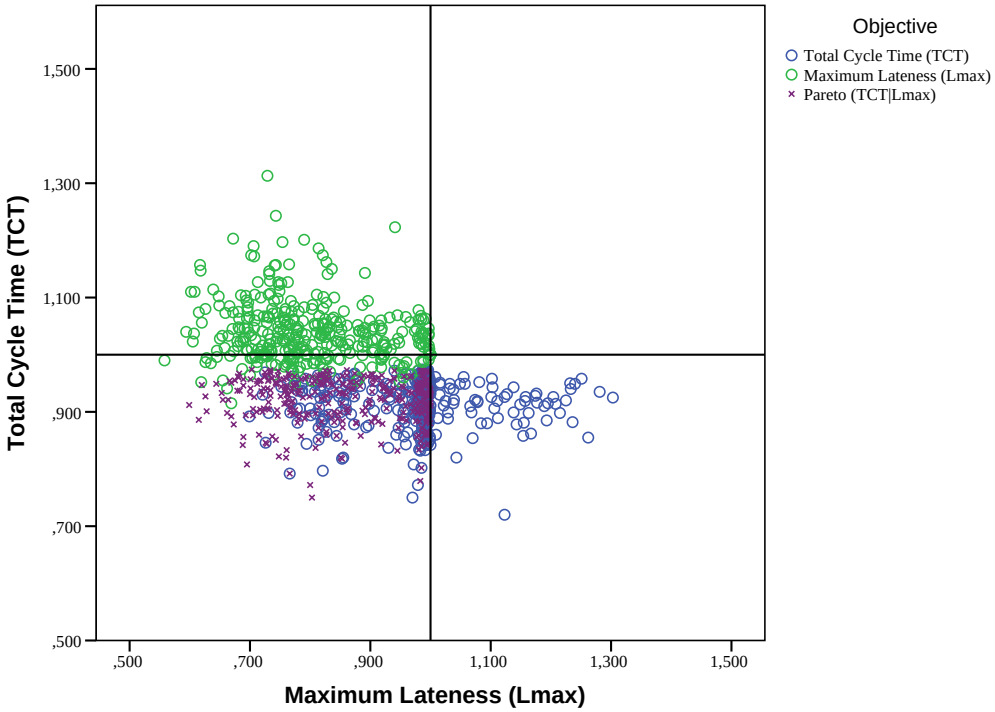
Experiment E9: pareto optimization of makespan and total tardiness



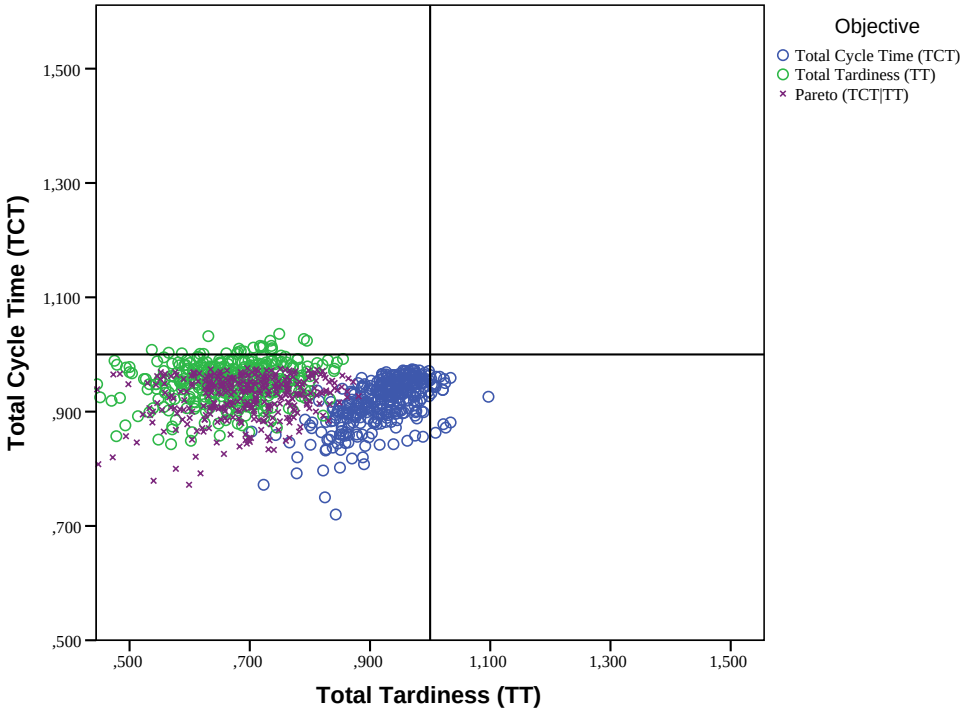
Experiment E9: pareto optimization of makespan and total unit penalties



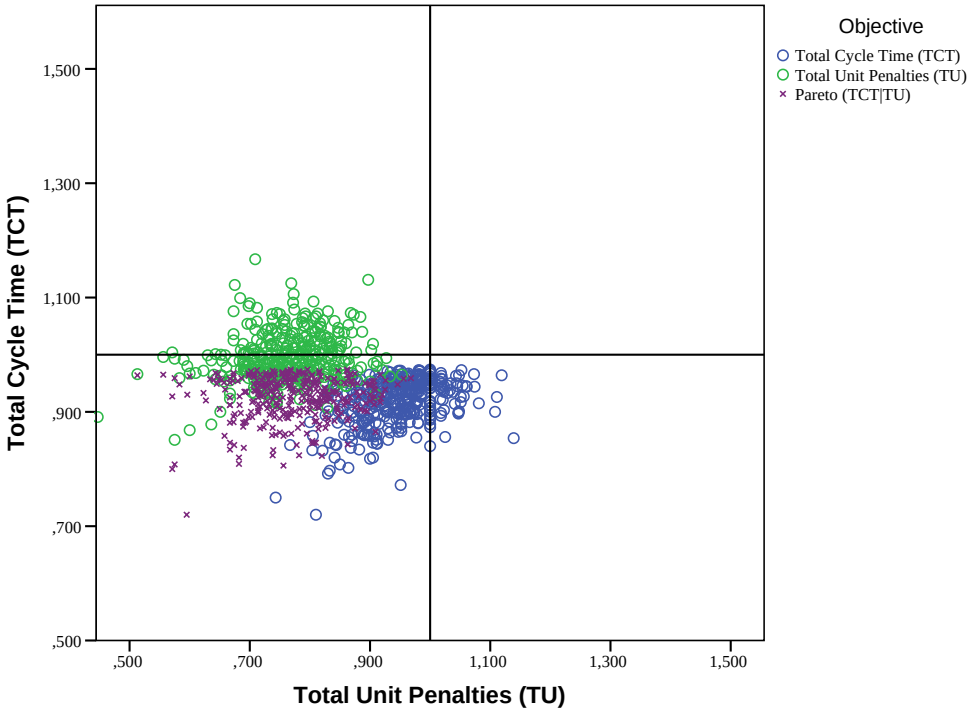
Experiment E9: pareto optimization of total cycle time and makespan



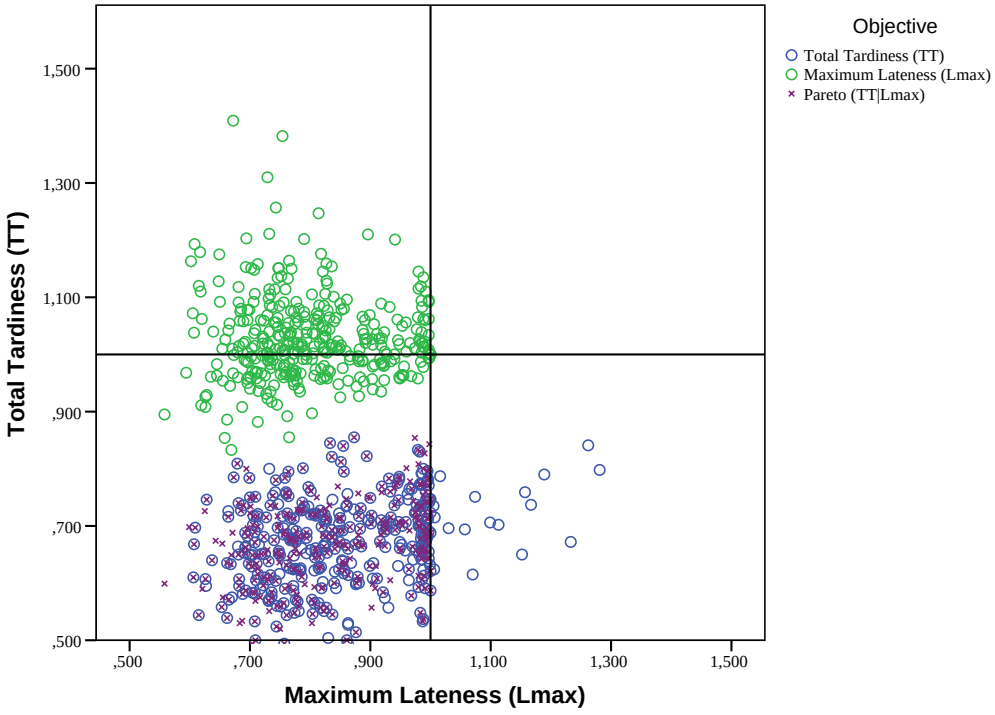
Experiment E9: pareto optimization of total cycle time and maximum lateness



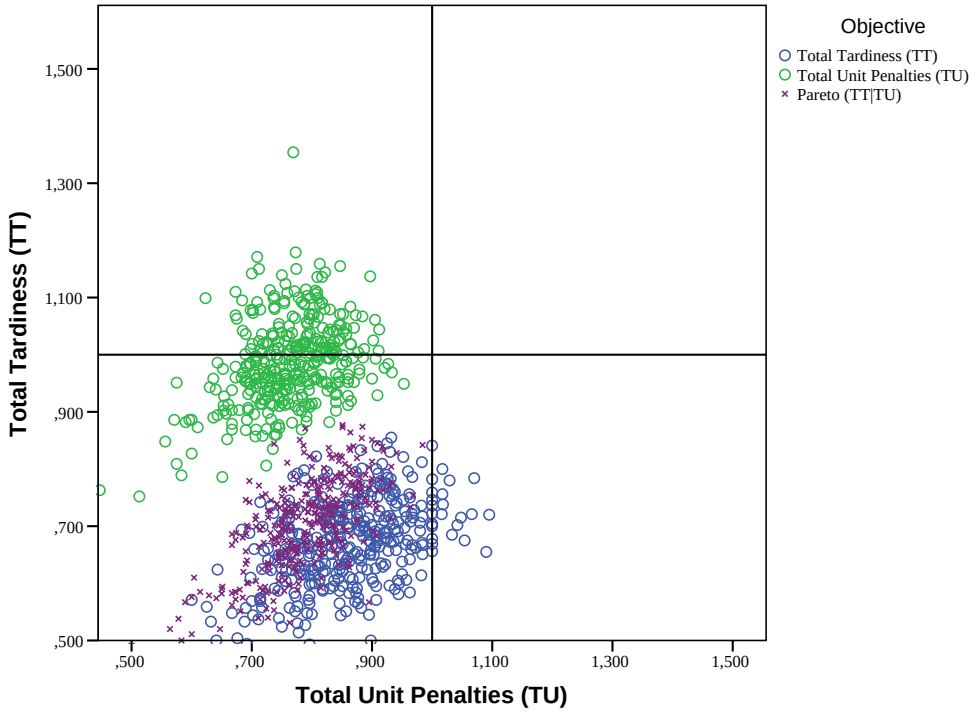
Experiment E9: pareto optimization of total cycle time and total tardiness



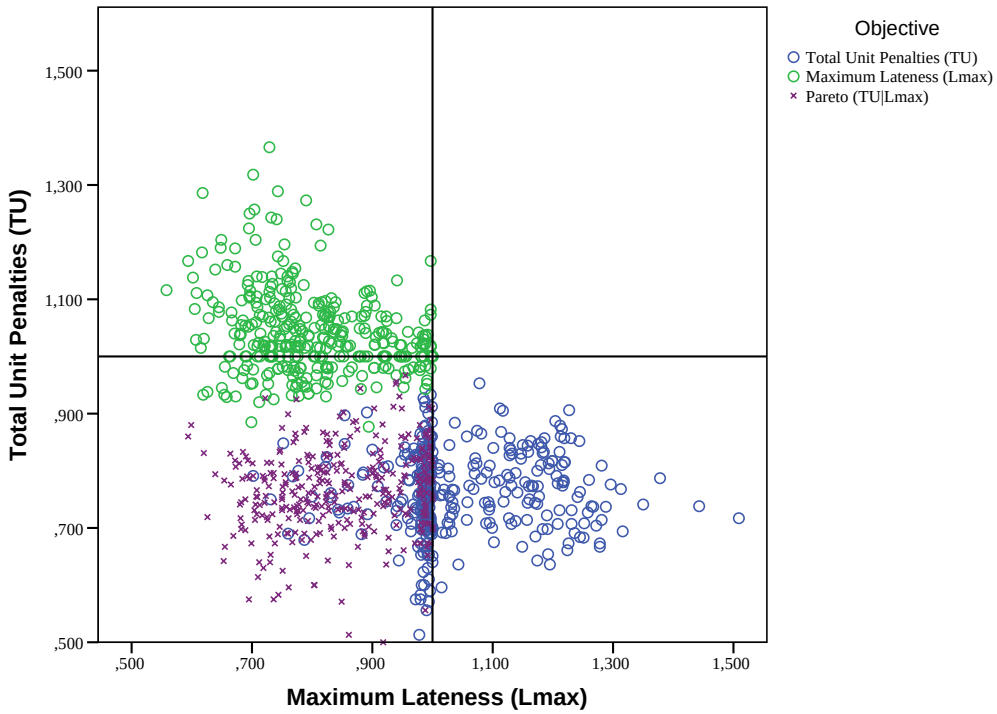
Experiment E9: pareto optimization of total cycle time and total unit penalties



Experiment E9: pareto optimization of total tardiness and maximum lateness



Experiment E9: pareto optimization of total tardiness and total unit penalties

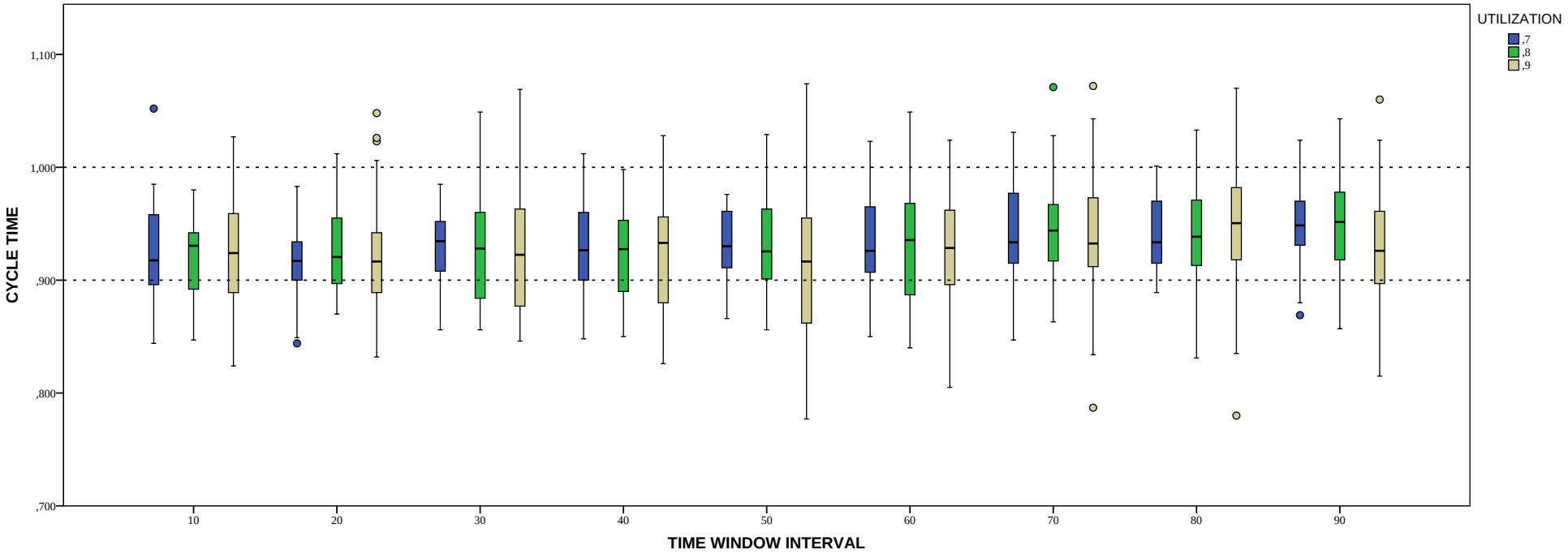


Experiment E9: pareto optimization of total unit penalties and maximum lateness

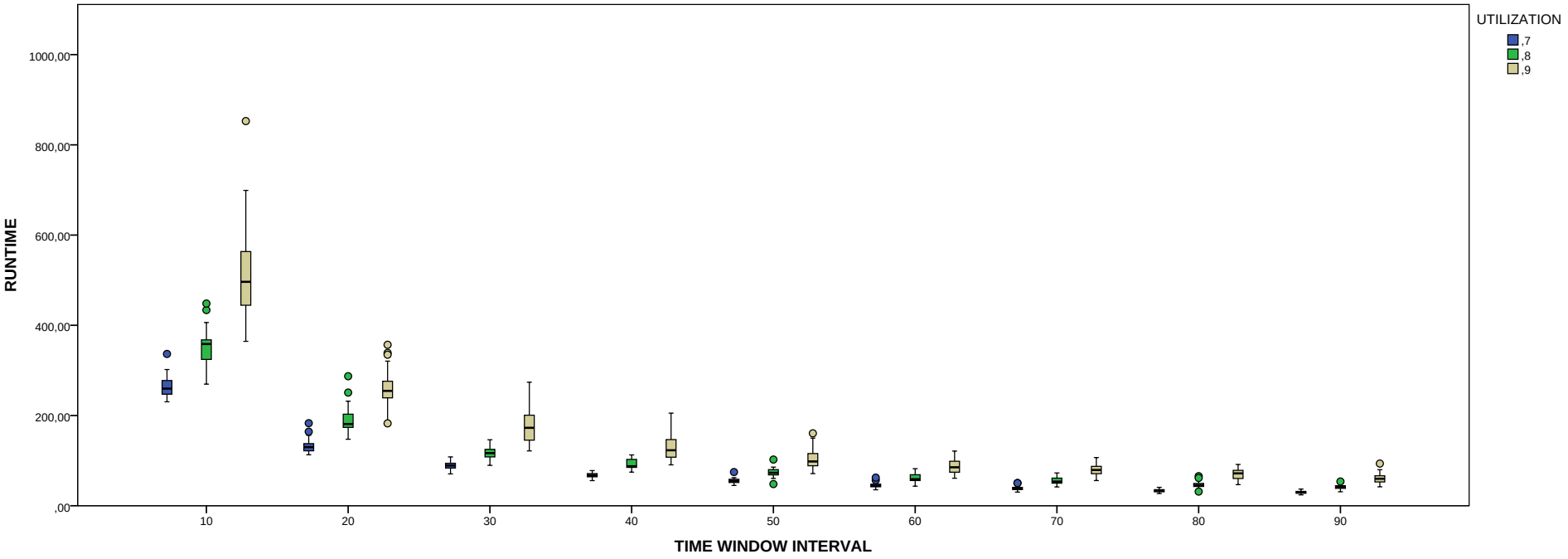
E.10 Experiment E.10

Design of experiment E.10

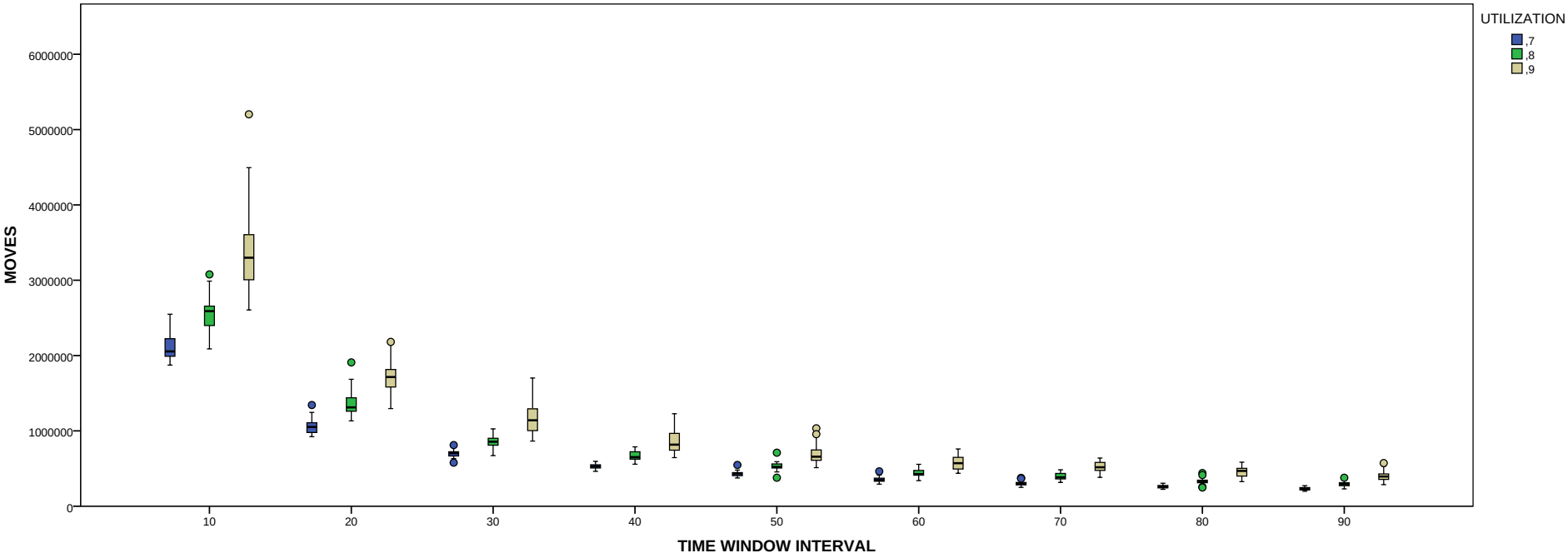
factor	level	count
machines	8	1
jobs	480	1
utilization	0.7, 0.8, 0.9	3
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	-	1
job arrivals	$U(0, C_{\max})$	1
job arrival errors	-	1
problem parameter combinations		3
model instances per parameter combination		30
total number of model instances		90
initial solution	FIFO	1
objective(s)	TCT	1
VNS type	VND	1
deadline	30 sec	1
time window interval	10, 20, 30, ..., 90 min	9
look-ahead horizon	90	1
number of methods and settings		9
replications per run		1
total number of runs		810



Experiment E10: cycle time improvements depending on interval length



Experiment E10: run time required for cycle time improvements depending on interval length

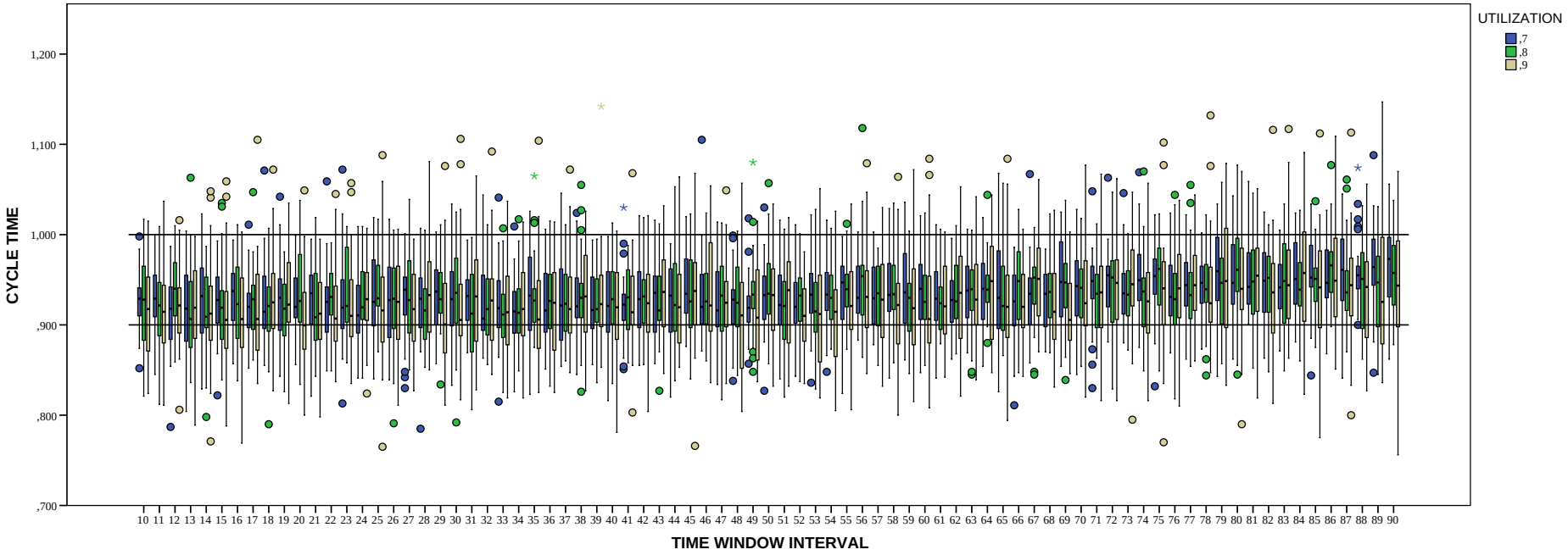


Experiment E10: moves required for cycle time improvements depending on interval length

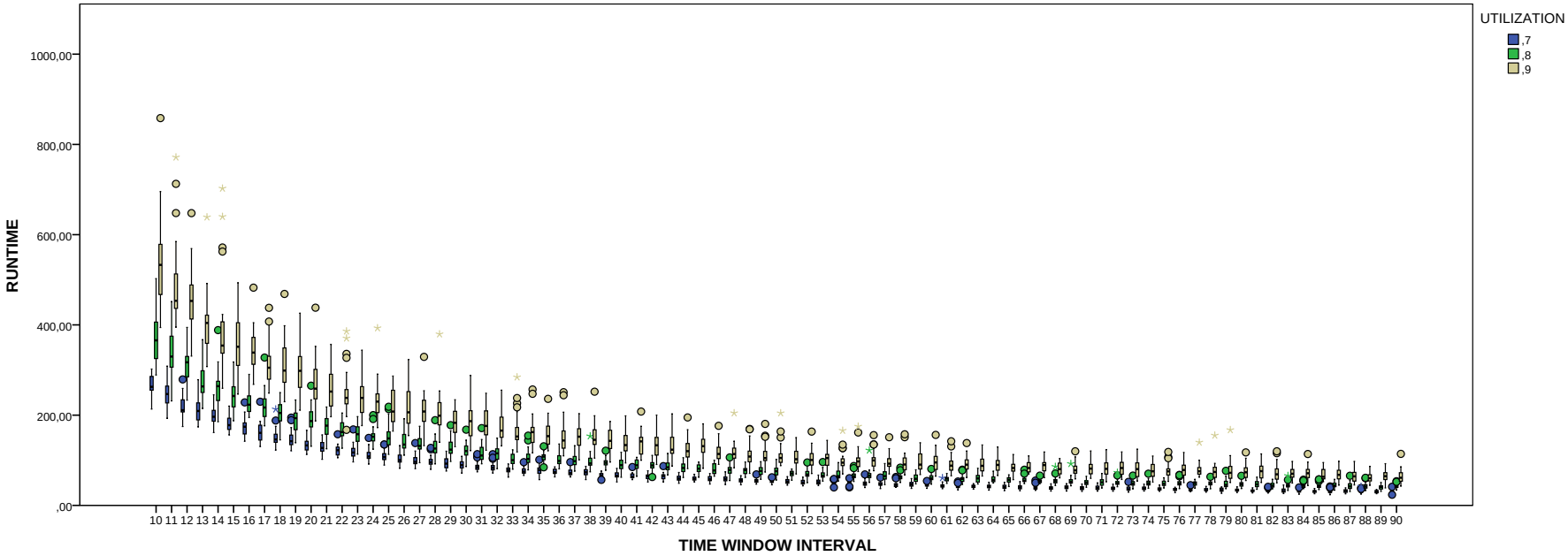
E.11 Experiment E.11

Design of experiment E.11

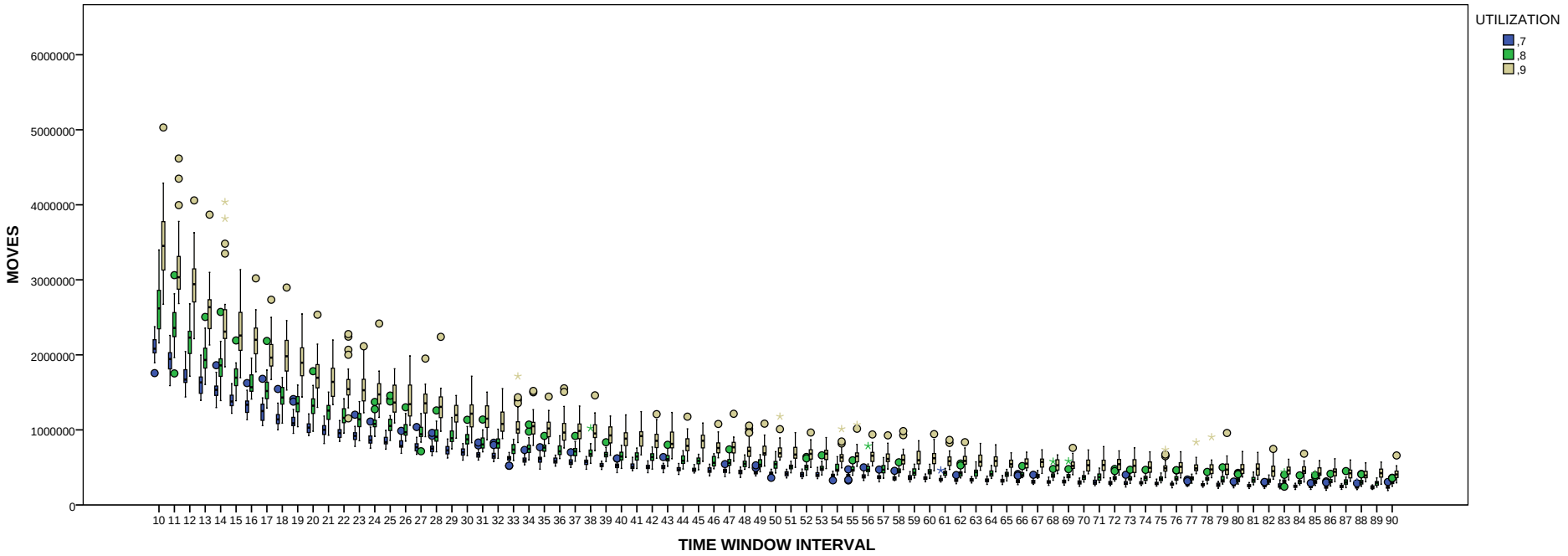
factor	level	count
machines	8	1
jobs	480	1
utilization	0.7, 0.8, 0.9	3
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	-	1
job arrivals	$U(0, C_{\max})$	1
job arrival errors	-	1
problem parameter combinations		3
model instances per parameter combination		30
total number of model instances		90
initial solution	FIFO	1
objective(s)	TCT	1
VNS type	VND	1
deadline	30 sec	1
time window interval	10, 11, 12, ..., 90 min	81
look-ahead horizon	90min	1
number of methods and settings		81
replications per run		1
total number of runs		7290



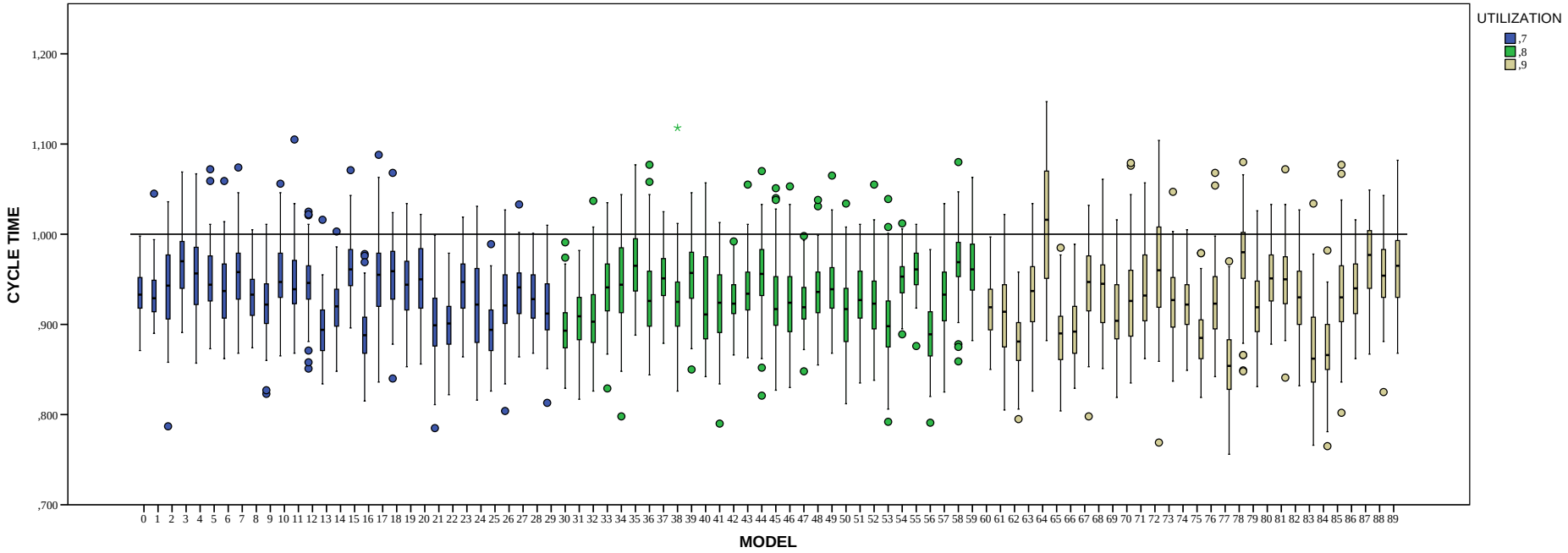
Experiment E11: cycle time improvements depending on interval length



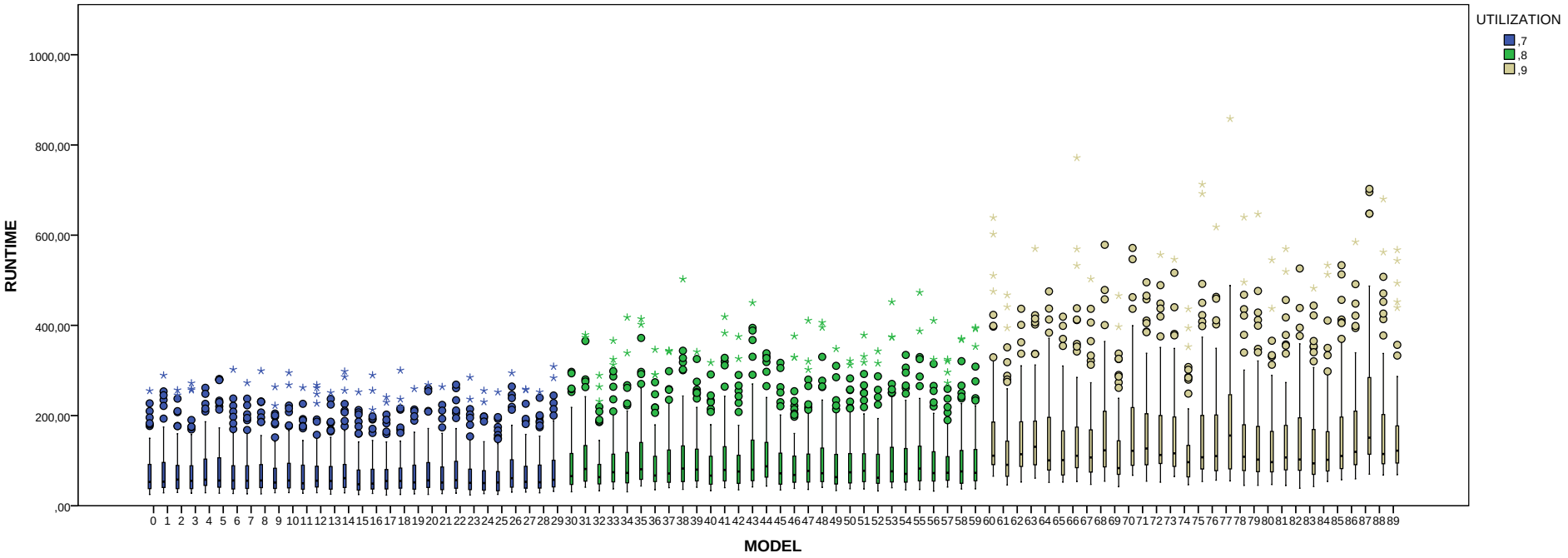
Experiment E11: run time required for cycle time improvements depending on interval length



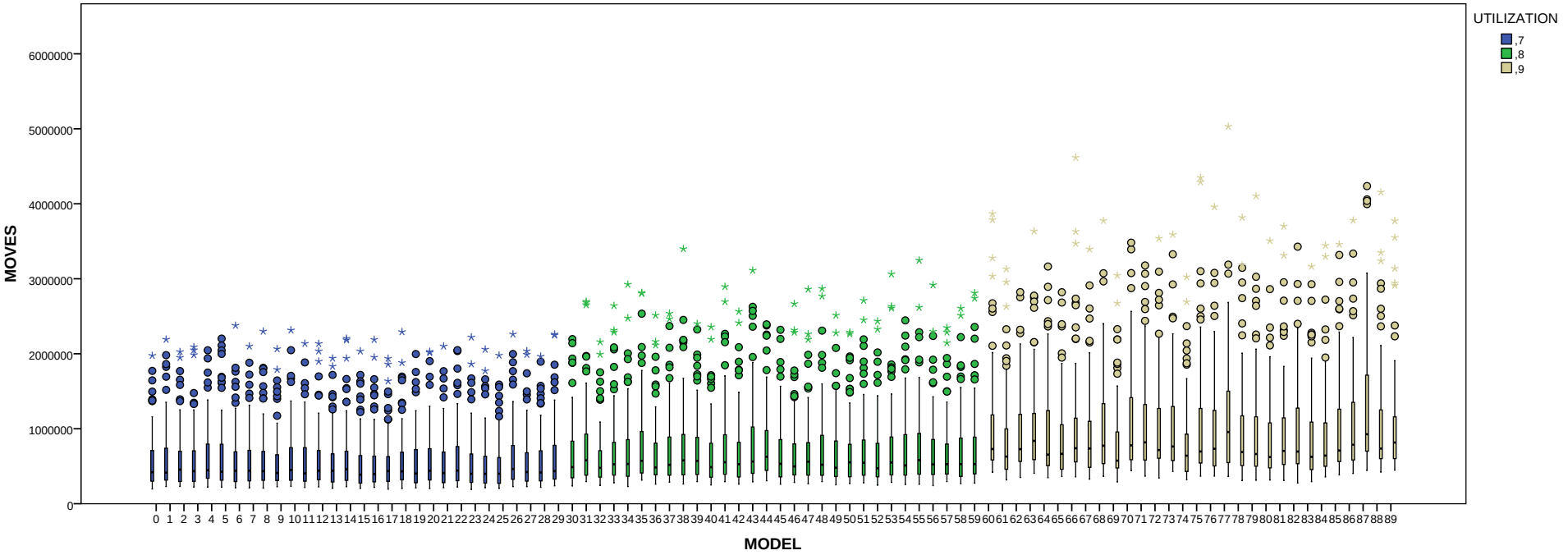
Experiment E11: moves required for cycle time improvements depending on interval length



Experiment E11: cycle time improvements depending on the model



Experiment E11: run time required for cycle time improvements depending on the model

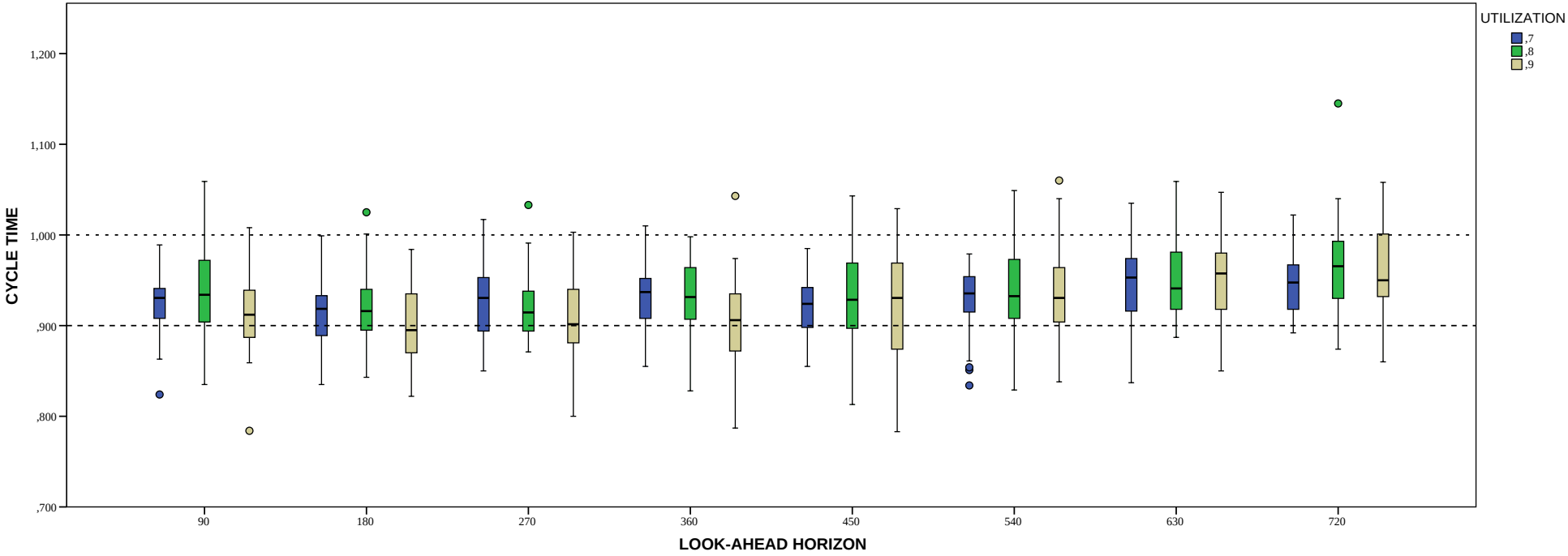


Experiment E11: moves required for cycle time improvements depending on the model

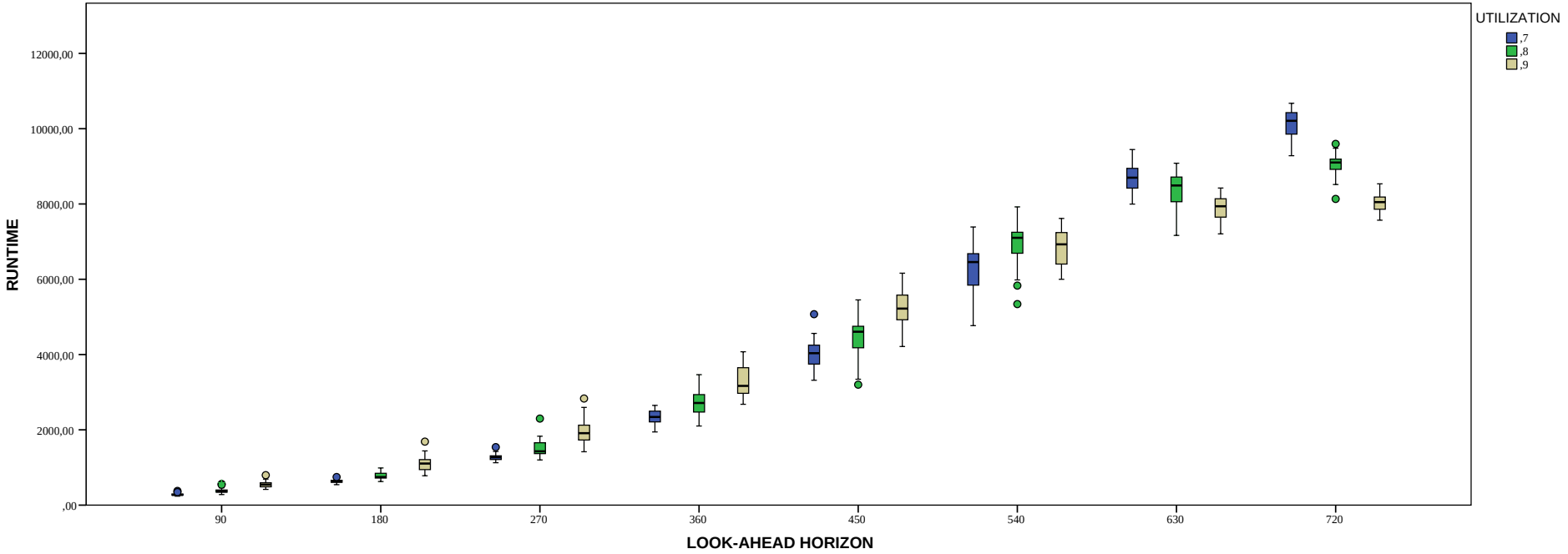
E.12 Experiment E.12

Design of experiment E.12

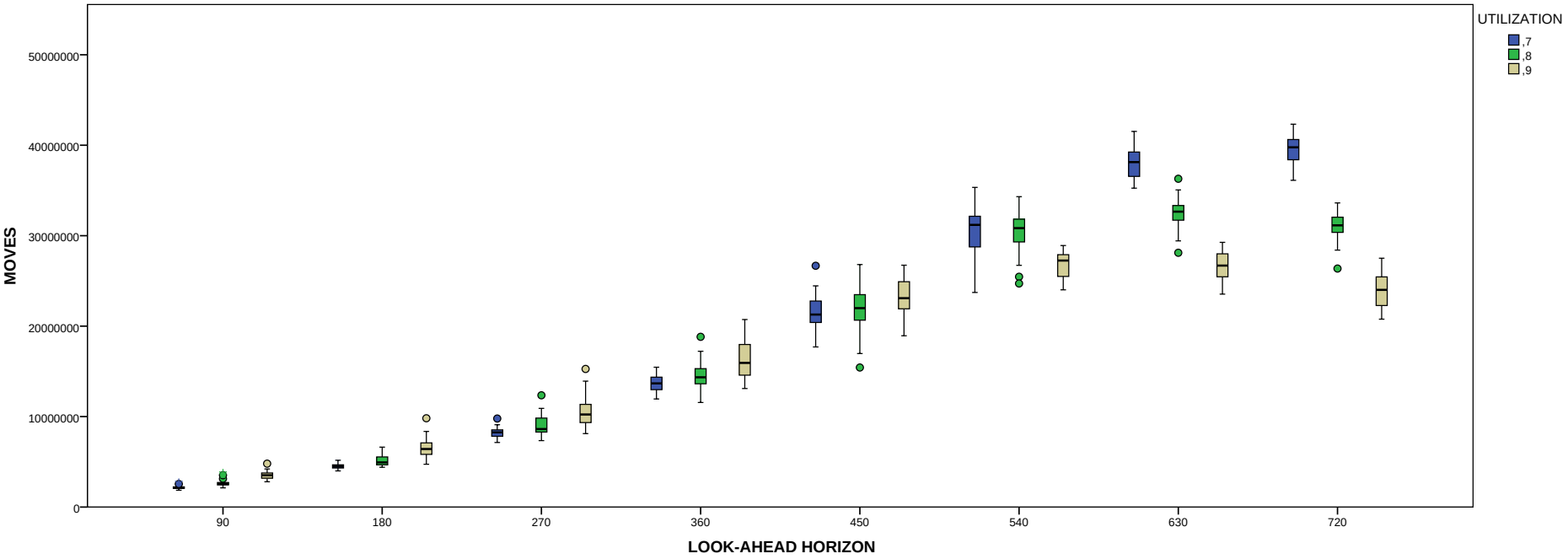
factor	level	count
machines	8	1
jobs	480	1
utilization	0.7, 0.8, 0.9	3
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	-	1
job arrivals	$U(0, C_{\max})$	1
job arrival errors	-	1
problem parameter combinations		3
model instances per parameter combination		30
total number of model instances		90
initial solution	FIFO	1
objective(s)	TCT	1
VNS type	VND	1
deadline	30 sec	1
time window interval	10	1
look-ahead horizon	90, 180, 270, ...,720	8
number of methods and settings		8
replications per run		1
total number of runs		720



Experiment E12: for cycle time improvements depending on the look-ahead horizon



Experiment E12: run time required for cycle time improvements depending on the look-ahead horizon

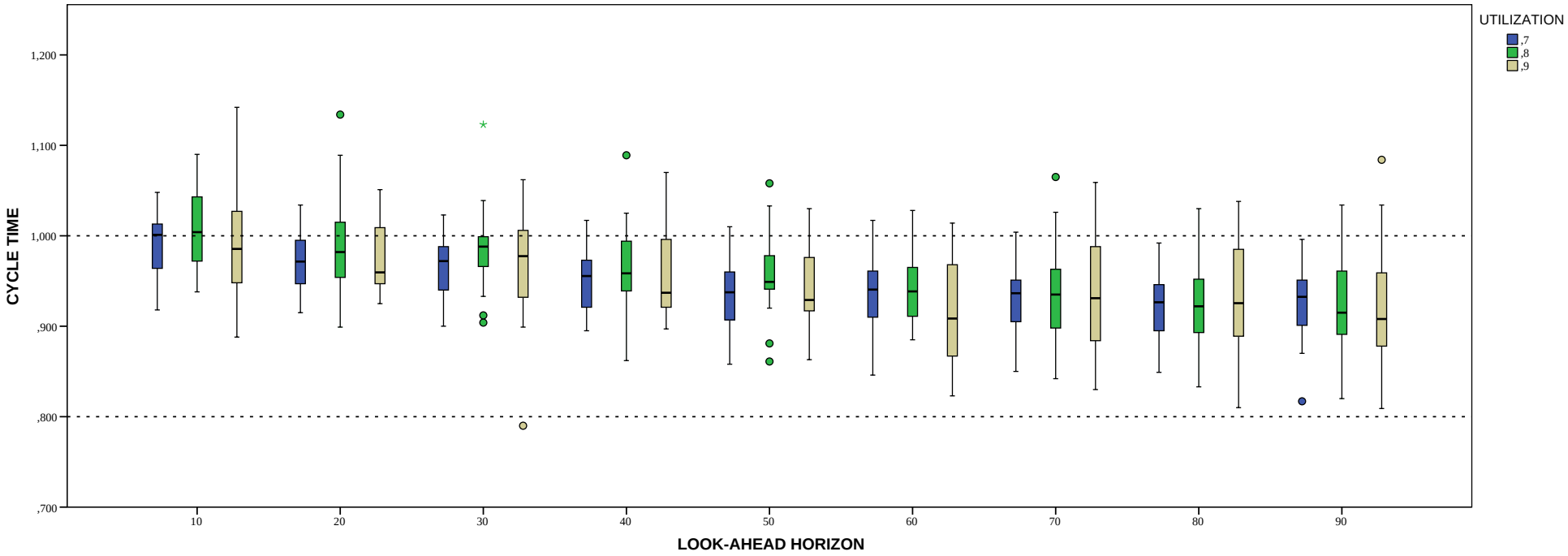


Experiment E12: moves required for cycle time improvements depending on the look-ahead horizon

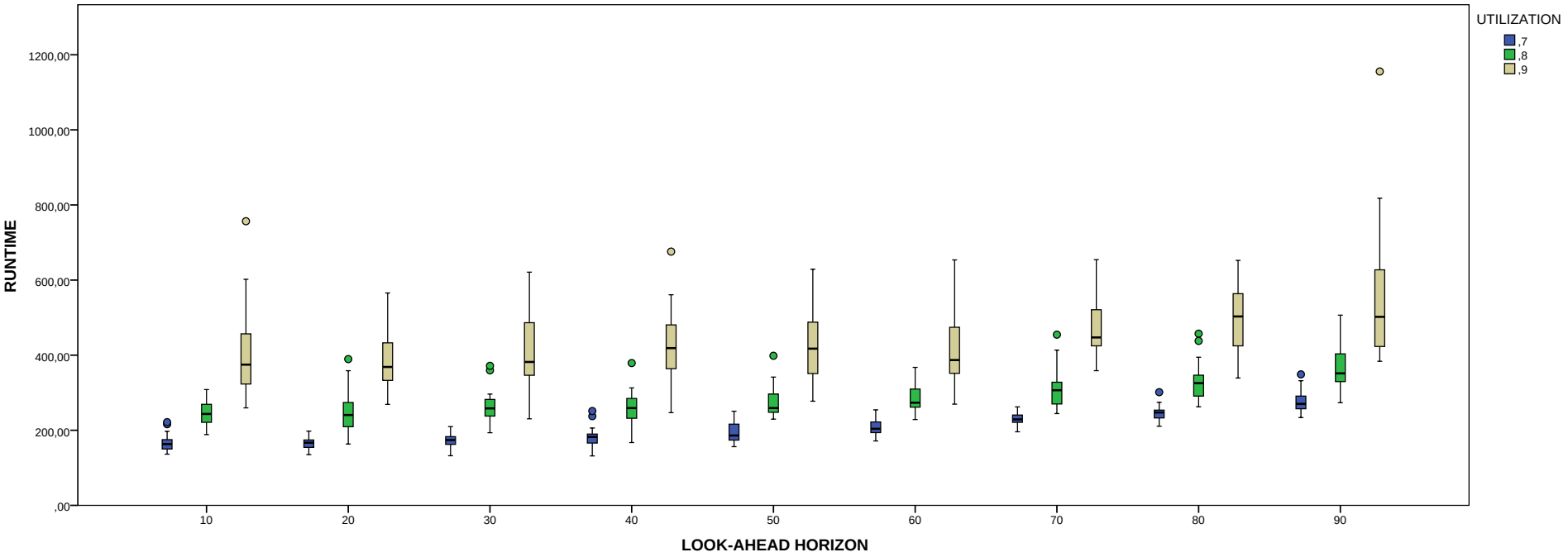
E.13 Experiment E.13

Design of experiment E.13

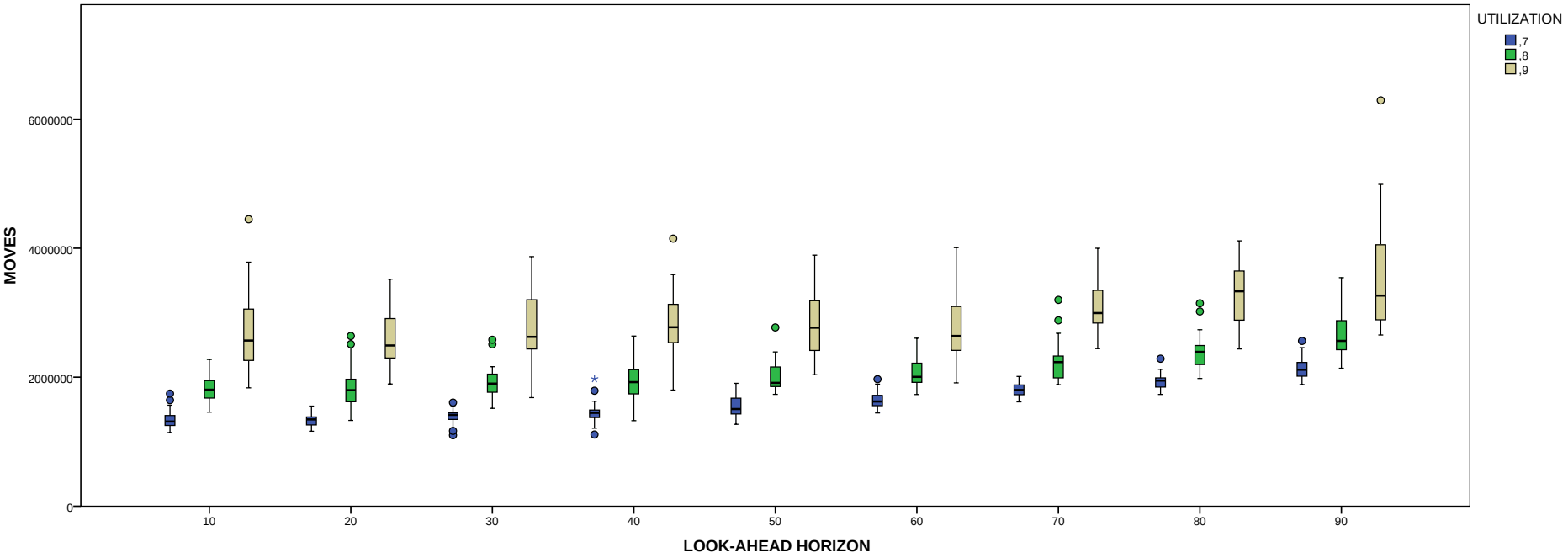
factor	level	count
machines	8	1
jobs	480	1
utilization	0.7, 0.8, 0.9	3
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	-	1
job arrivals	$U(0, C_{\max})$	1
job arrival errors	-	1
problem parameter combinations		3
model instances per parameter combination		30
total number of model instances		90
initial solution	FIFO	1
objective(s)	TCT	1
VNS type	VND	1
deadline	30 sec	1
time window interval	10	1
look-ahead horizon	10, 20, 30, ..., 90	9
number of methods and settings		9
replications per run		1
total number of runs		810



Experiment E13: cycle time improvements depending on the look-ahead horizon



Experiment E13: run time required for cycle time improvements depending on the look-ahead horizon

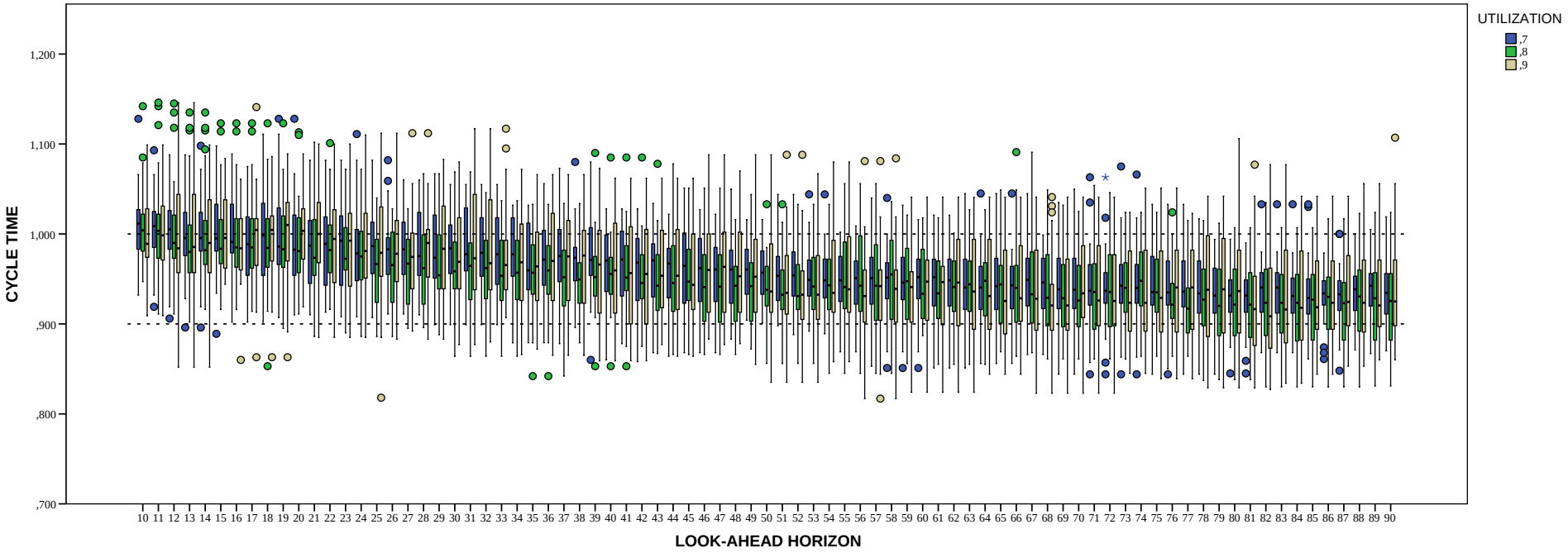


Experiment E13: moves required for cycle time improvements depending on the look-ahead horizon

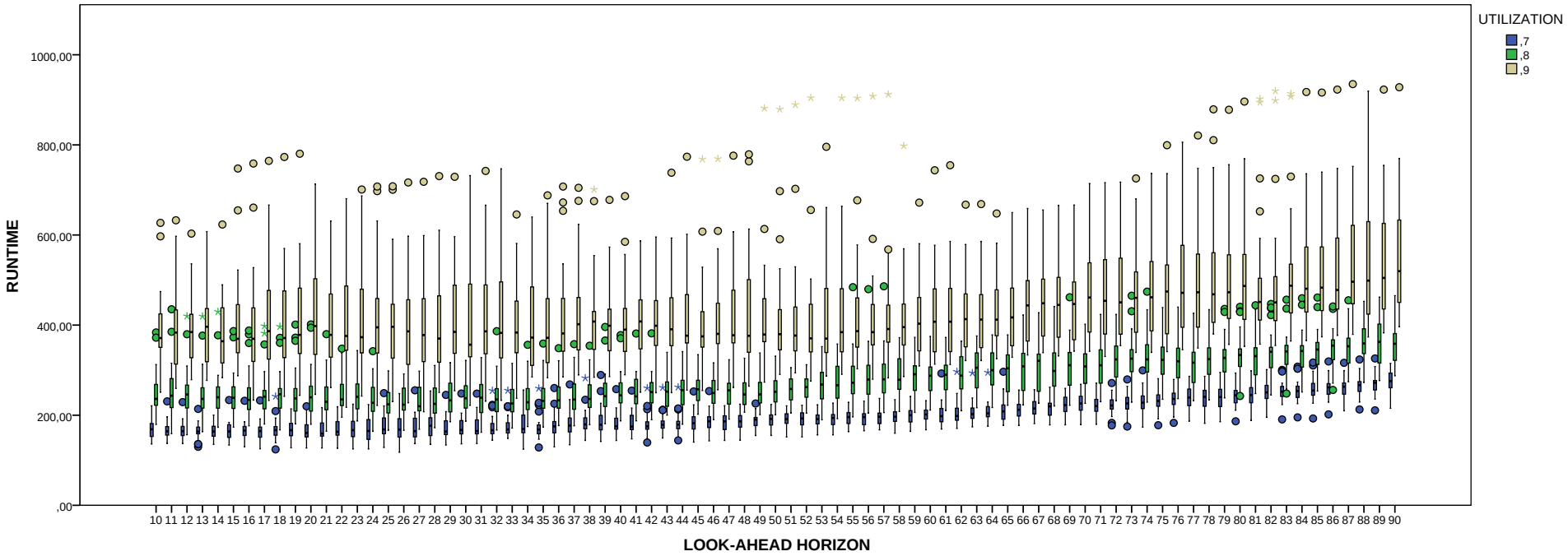
E.14 Experiment E.14

Design of experiment E.14

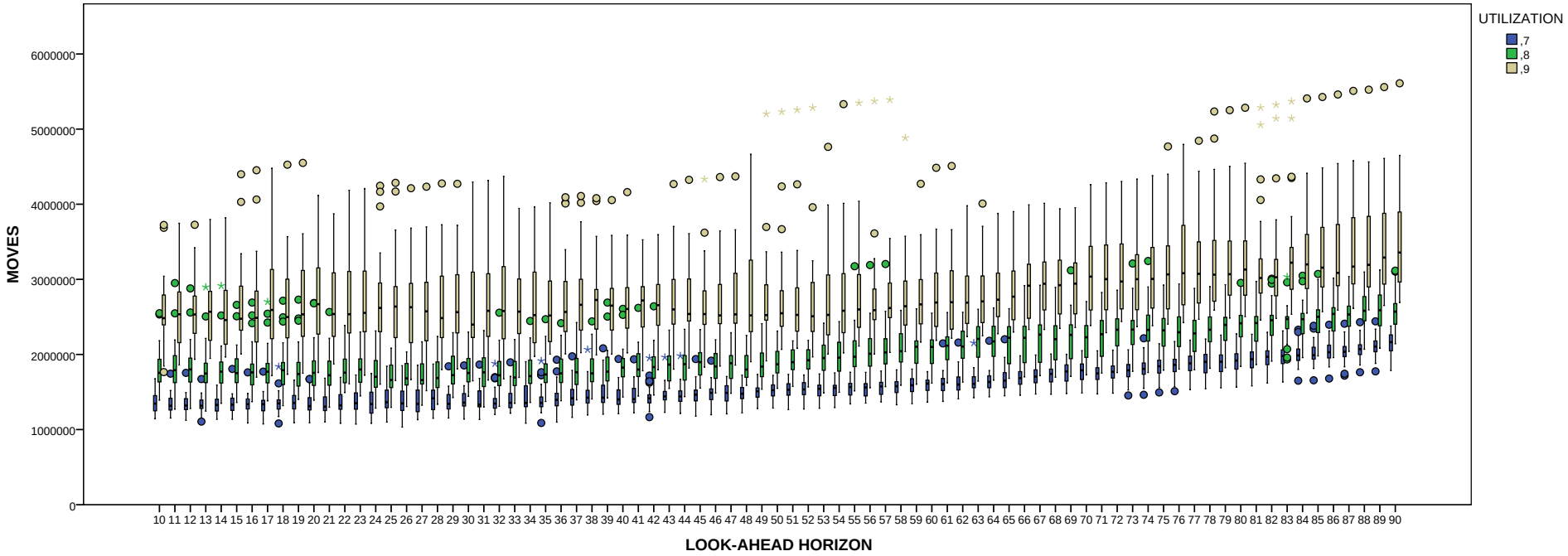
factor	level	count
machines	8	1
jobs	480	1
utilization	0.7, 0.8, 0.9	3
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	-	1
job arrivals	$U(0, C_{\max})$	1
job arrival errors	-	1
problem parameter combinations		3
model instances per parameter combination		30
total number of model instances		90
initial solution	FIFO	1
objective(s)	TCT	1
VNS type	VND	1
deadline	30 sec	1
time window interval	10	1
look-ahead horizon	1, 2, 3, ..., 90	81
number of methods and settings		81
replications per run		1
total number of runs		7290



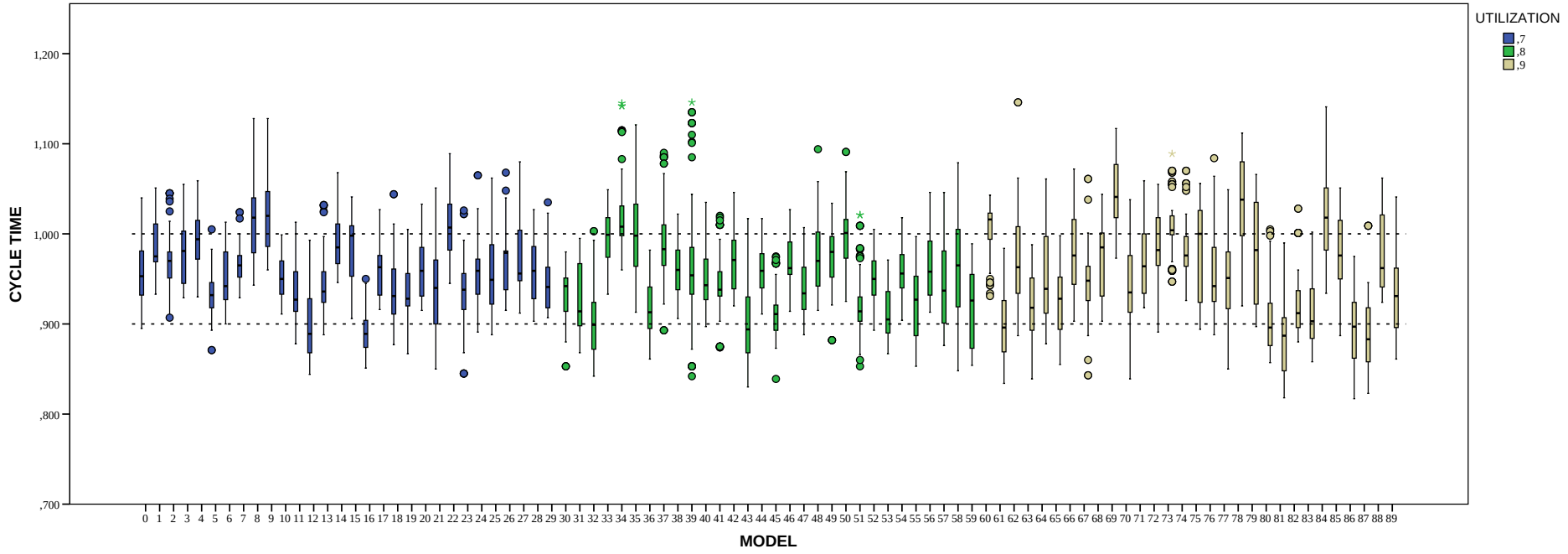
Experiment E14: cycle time improvements depending on the look-ahead horizon



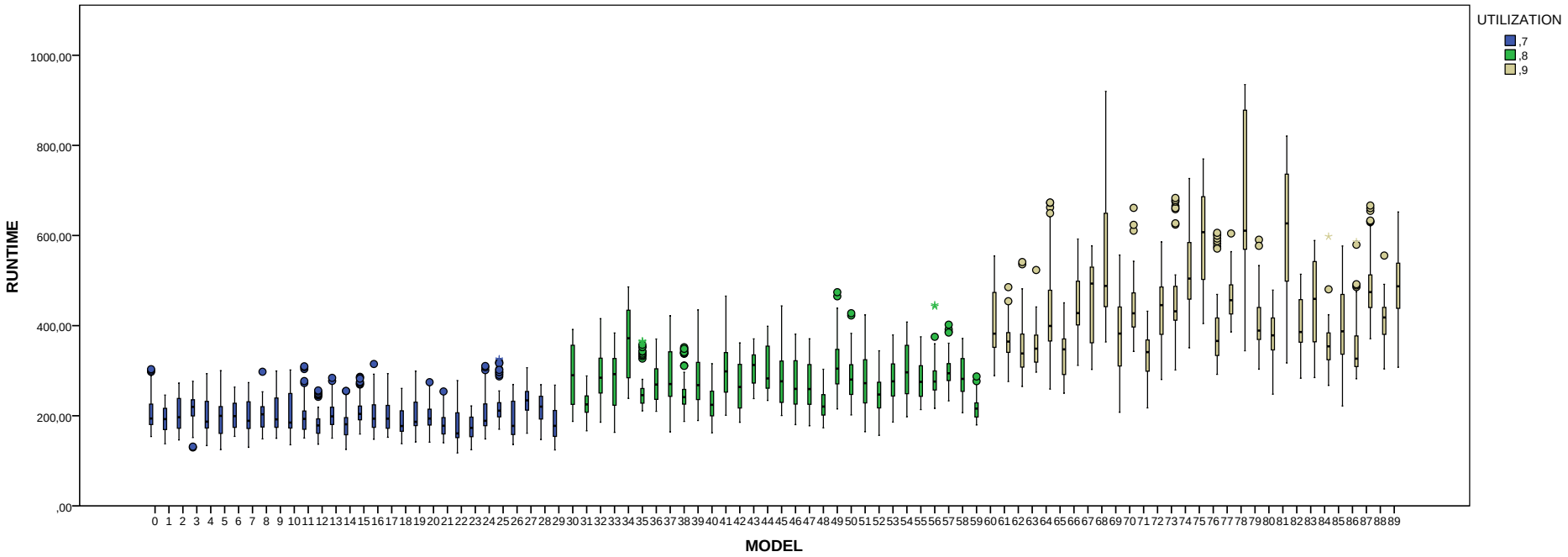
Experiment E14: run time required for cycle time improvements depending on the look-ahead horizon



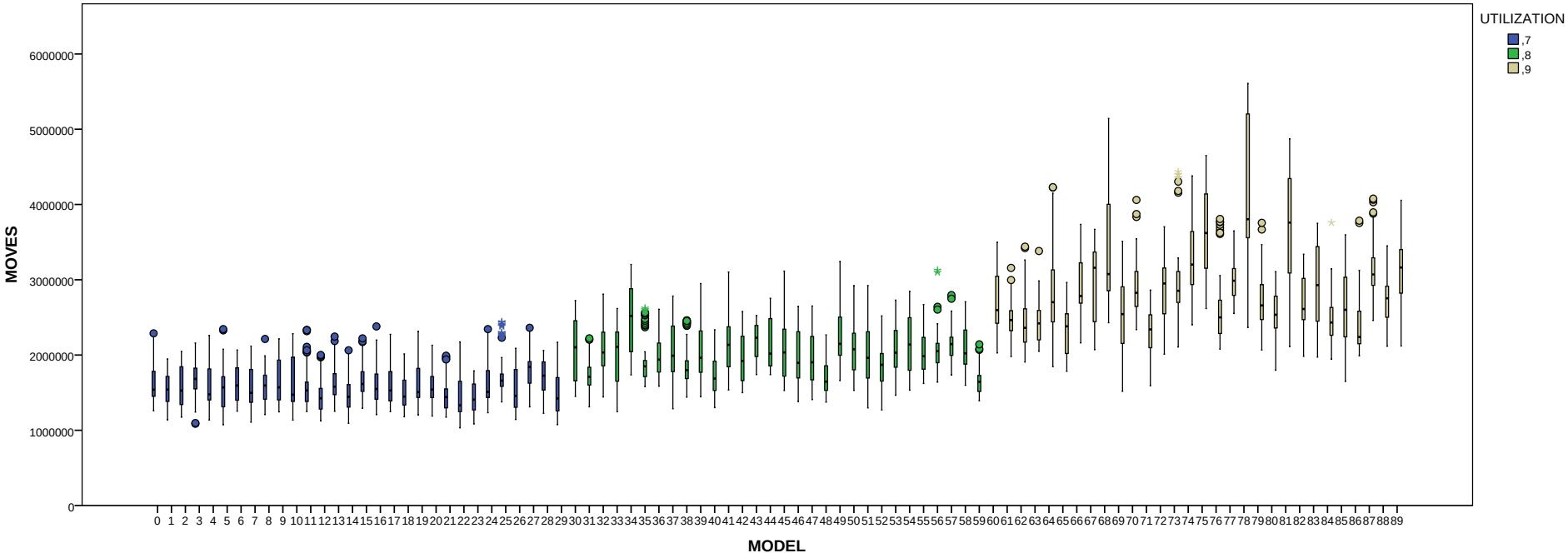
Experiment E14: moves required for cycle time improvements depending on the look-ahead horizon



Experiment E14: cycle time improvements depending on the look-ahead horizon



Experiment E14: run time required for cycle time improvements depending on the look-ahead horizon

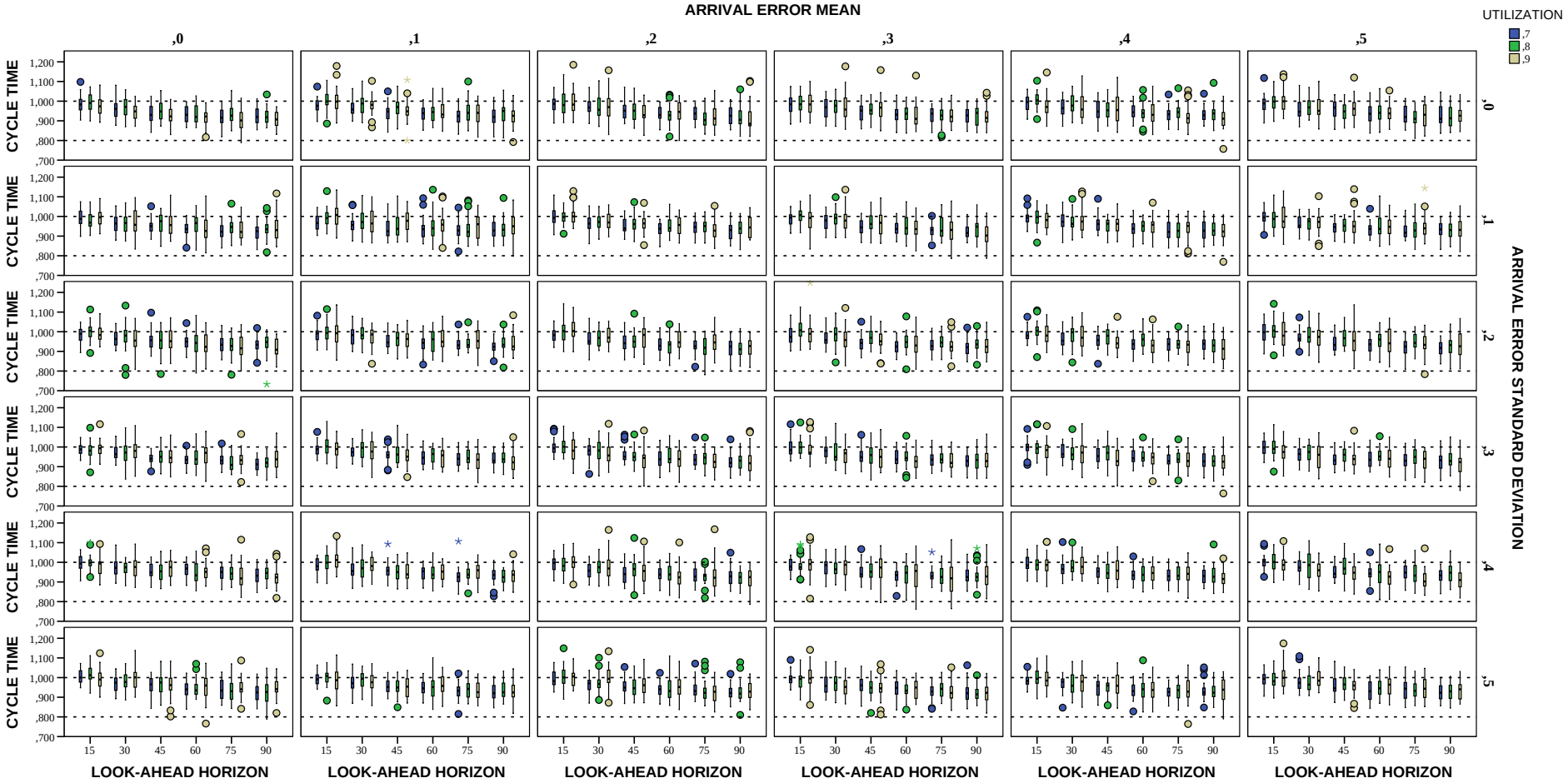


Experiment E14: moves required for cycle time improvements depending on the look-ahead horizon

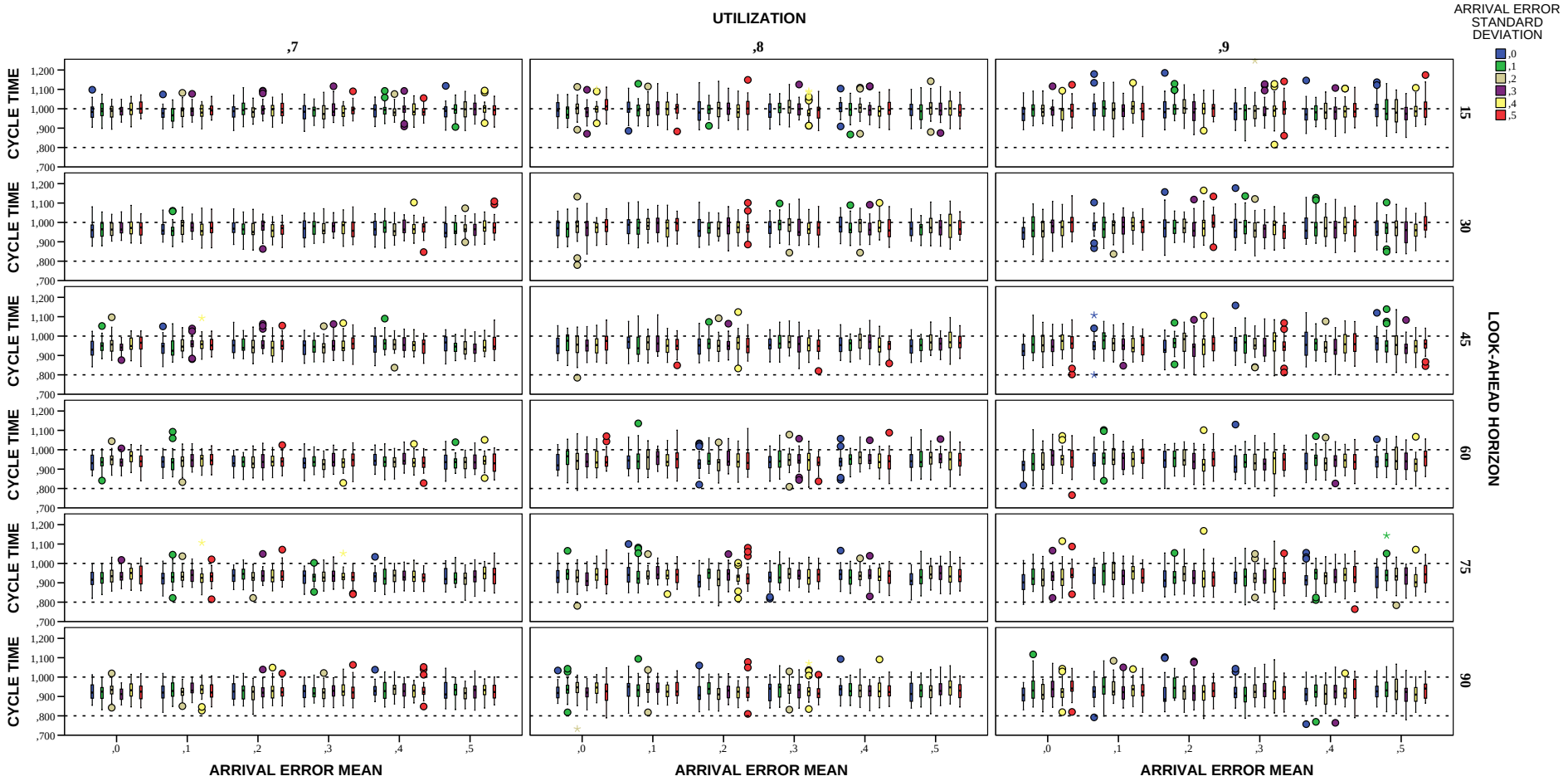
E.15 Experiment E.15

Design of experiment E.15

factor	level	count
machines	8	1
jobs	480	1
utilization	0.7, 0.8, 0.9	3
job families	8	1
dedication density	$1(uniform)$	1
processing times	$U(240, 480)$	1
batch size (lots)	8	1
batch size (wafers)	200	1
job sizes	$s_j = 25$	1
job priorities	-	1
job due dates	-	1
job arrivals	$U(0, C_{\max})$	1
job arrival errors	$N(mean, stdv)$	36
problem parameter combinations		108
model instances per parameter combination		30
total number of model instances		3240
initial solution	FIFO	1
objective(s)	TCT	1
VNS type	VND	1
deadline	30 sec	1
time window interval	10	1
look-ahead horizon	15, 30, 45, 60, 75, 90	6
number of methods and settings		6
replications per run		1
total number of runs		19440



Experiment E15: cycle time improvements depending on the errors in job arrival prediction (view1)

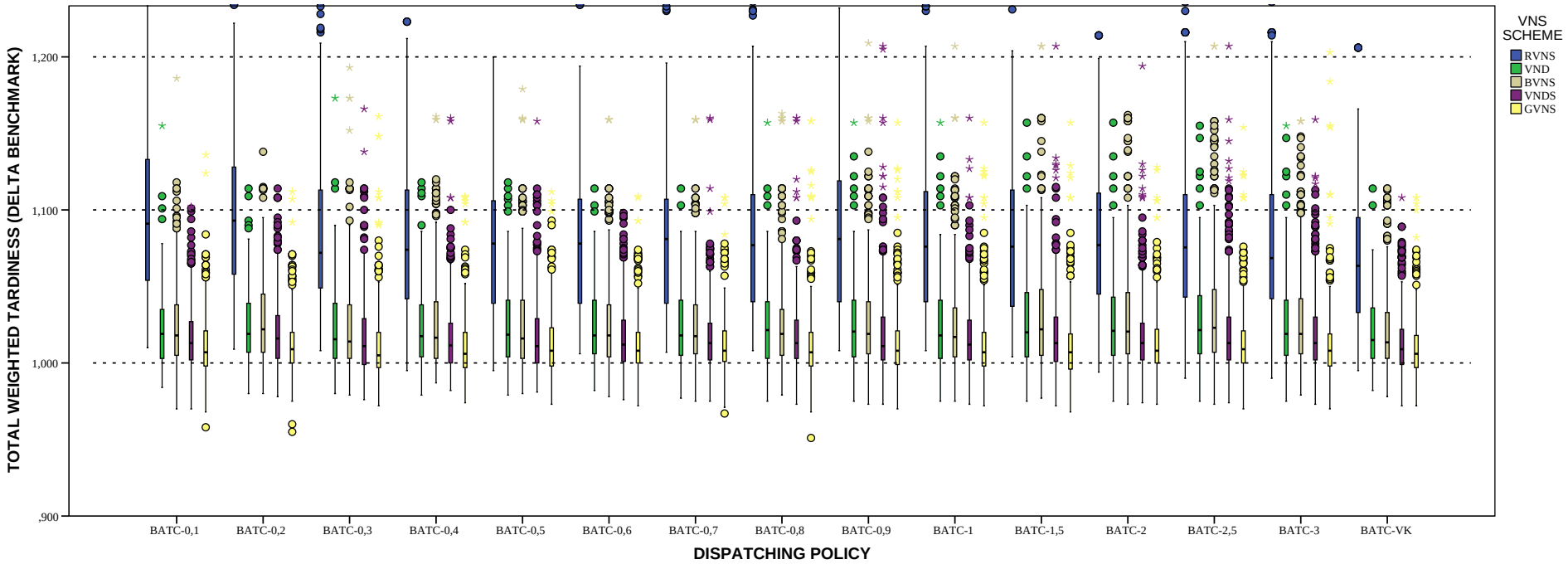


Experiment E15: cycle time improvements depending on the errors in job arrival prediction (view2)

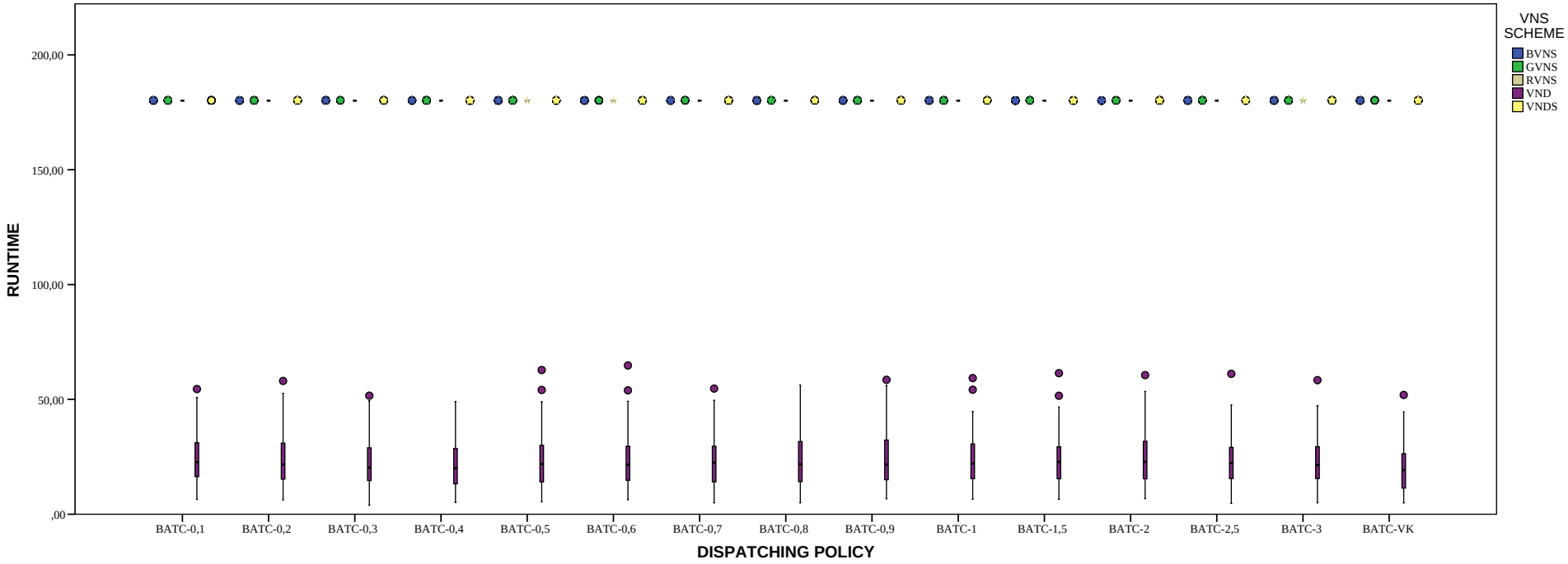
E.16 Experiment E.16

Design of experiment E.16

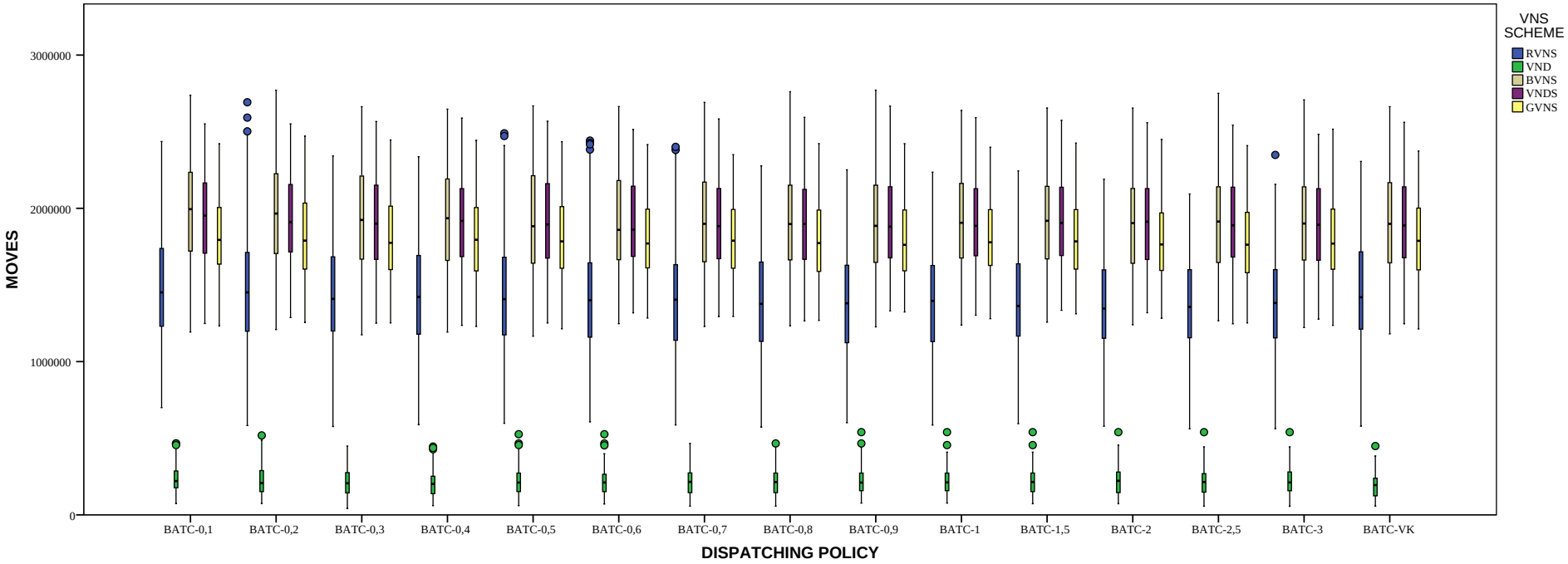
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	0.1, 0.2, 0.3, . . . , 1, 1.5, 2, 2.5, 3	15
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	VND, RVNS, BVNS, GVNS, VNDS	5
deadline [min]	3	1
neighborhoods	default neighborhood sequence	1
number of methods and settings		75
replications per run		5
total number of runs		33750



Experiment E16: total weighted tardiness improvements depending on the initial solution



Experiment E16: run time required for total weighted tardiness improvements depending on the initial solution

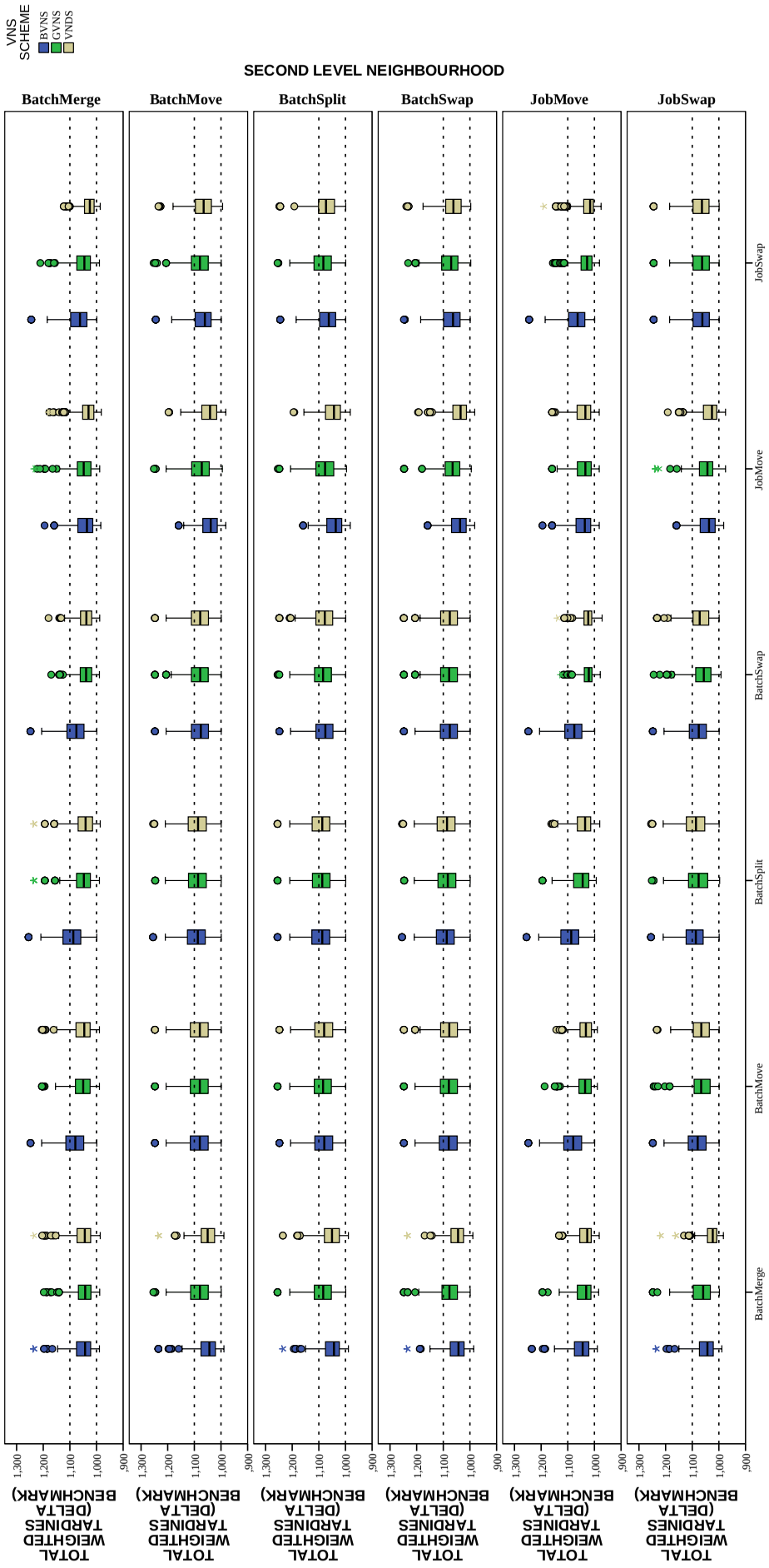


Experiment E16: moves required for total weighted tardiness improvements depending on the initial solution

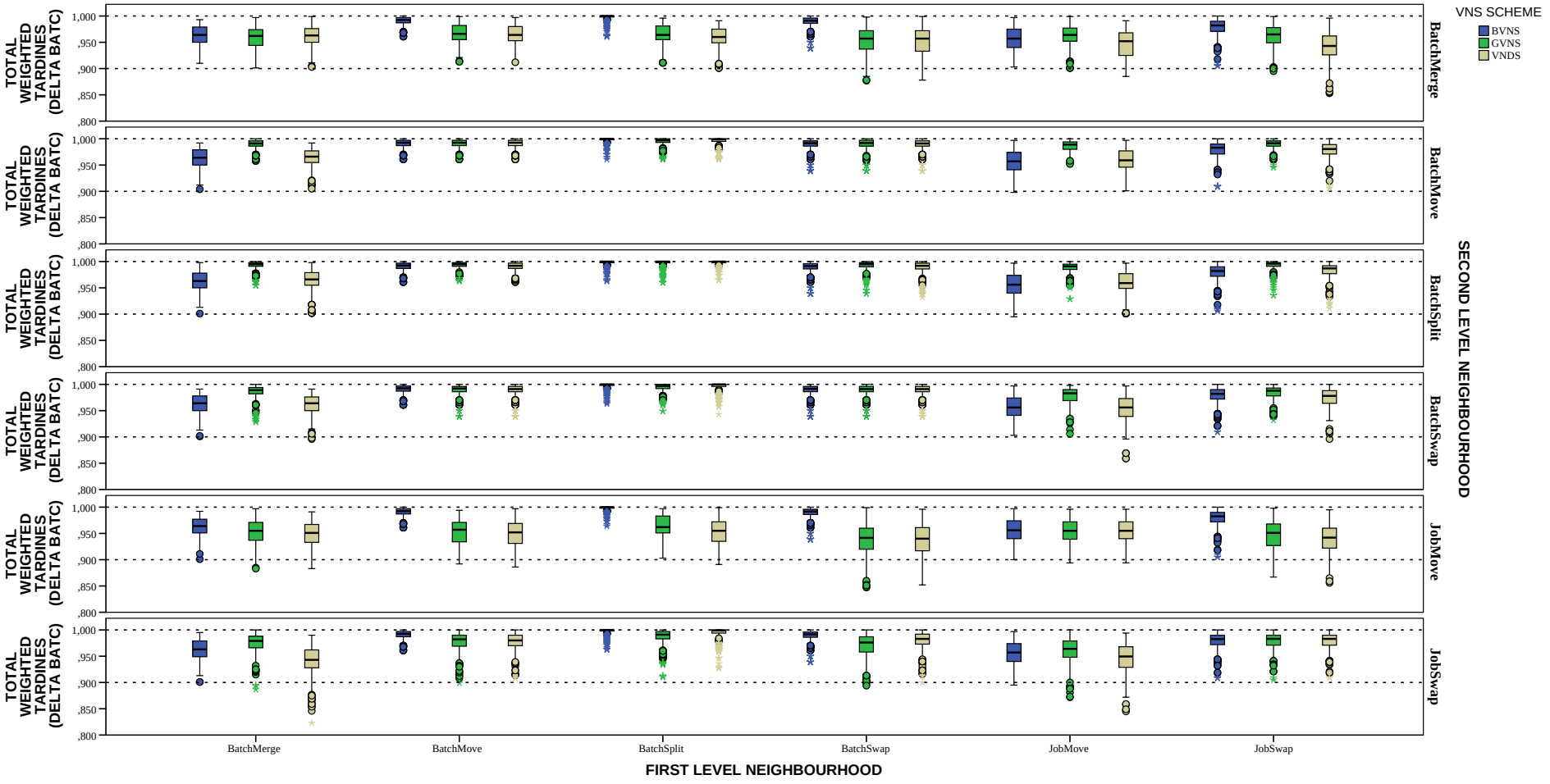
E.17 Experiment E.17

Design of experiment E.17

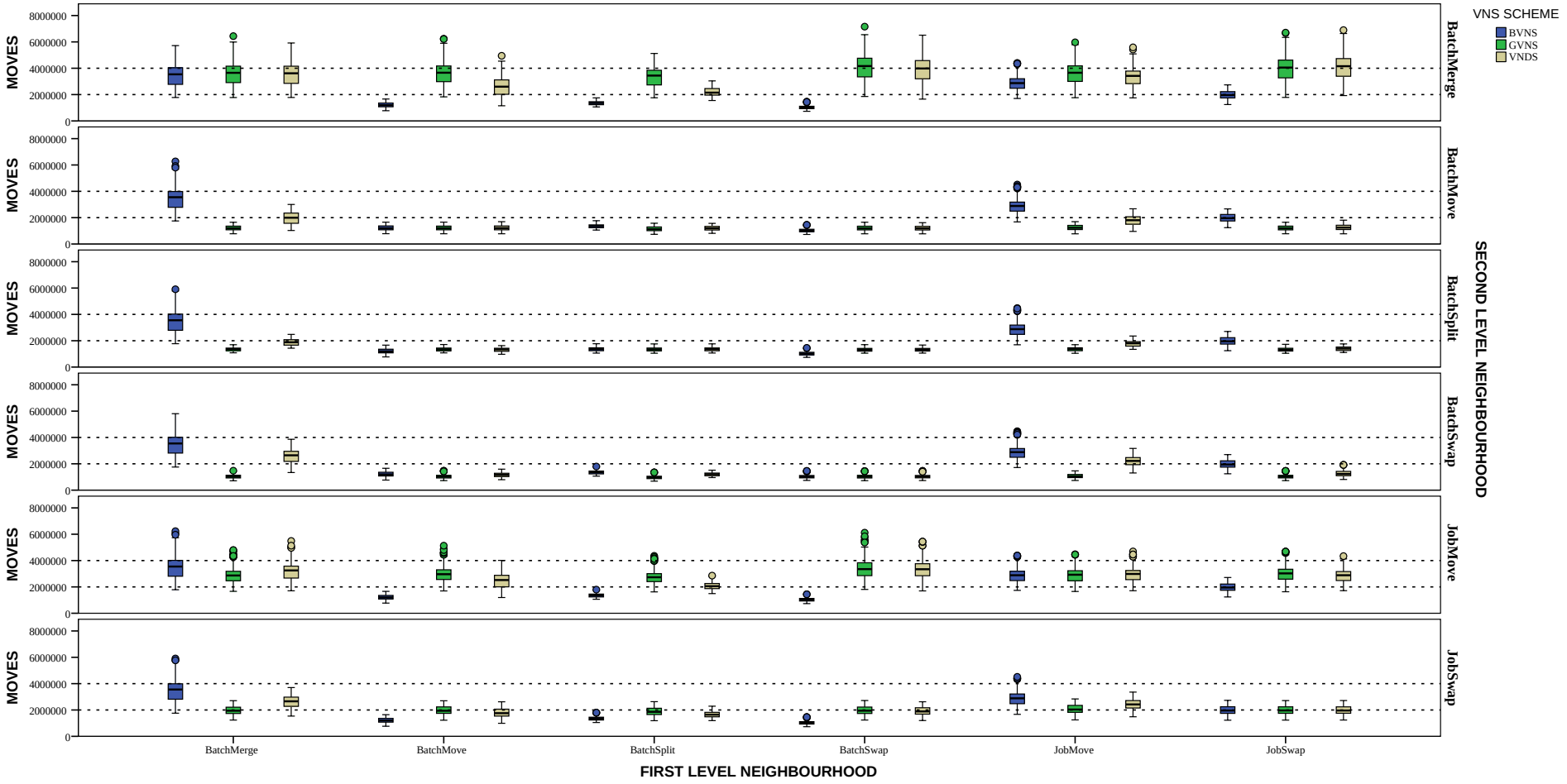
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	BVNS, GVNS, VNDS	3
deadline [min]	3	1
first level neighborhoods	six default neighborhoods	6
second level neighborhoods	six default neighborhoods	6
number of methods and settings		108
replications per run		5
total number of runs		48600



Experiment E17: total weighted tardiness improvements depending on the neighbourhood structure (Benchmark)



Experiment E17: total weighted tardiness improvements depending on the neighbourhood structure (BATC)

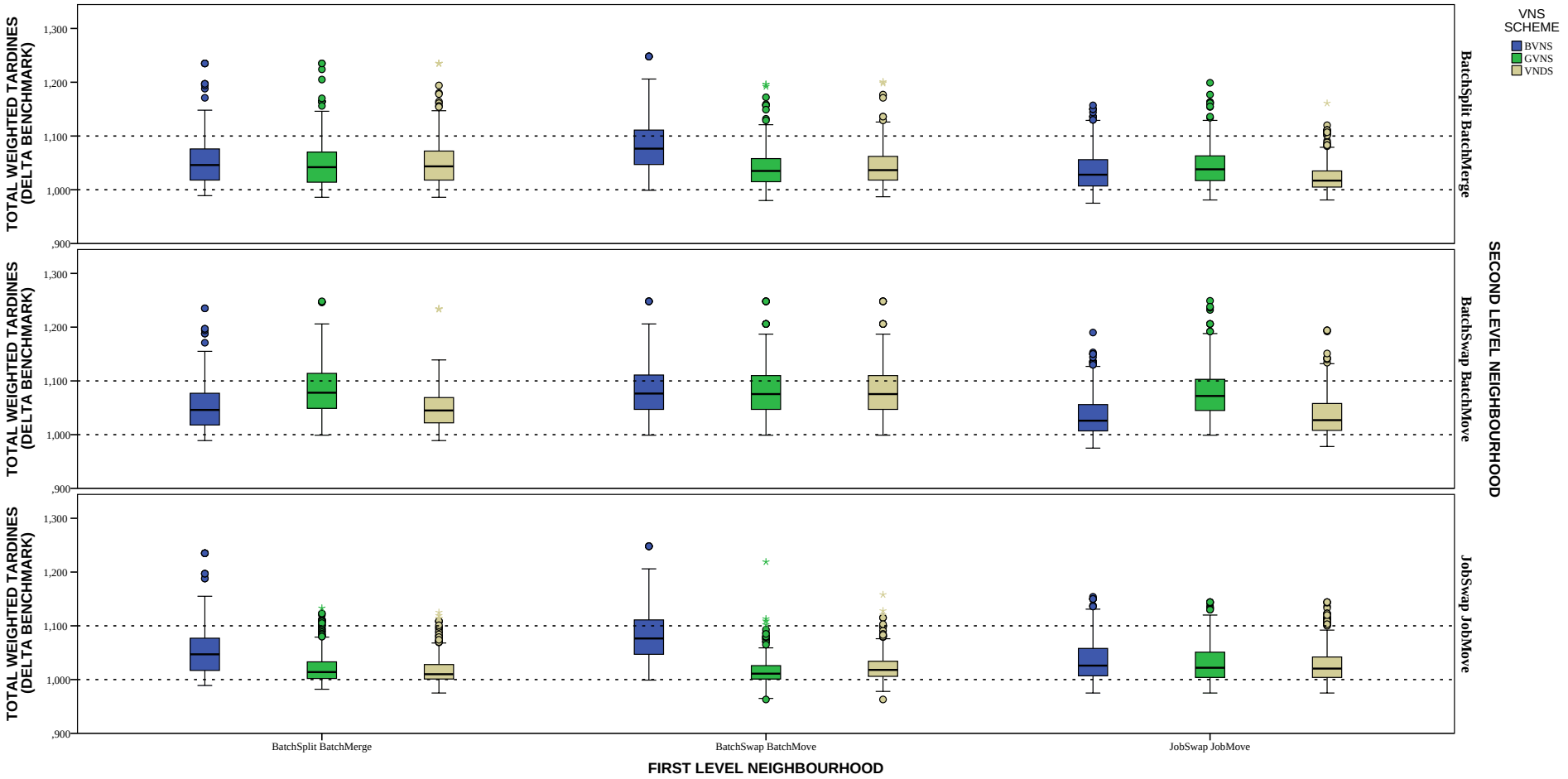


Experiment E17: moves required for total weighted tardiness improvements depending on the neighbourhood structure

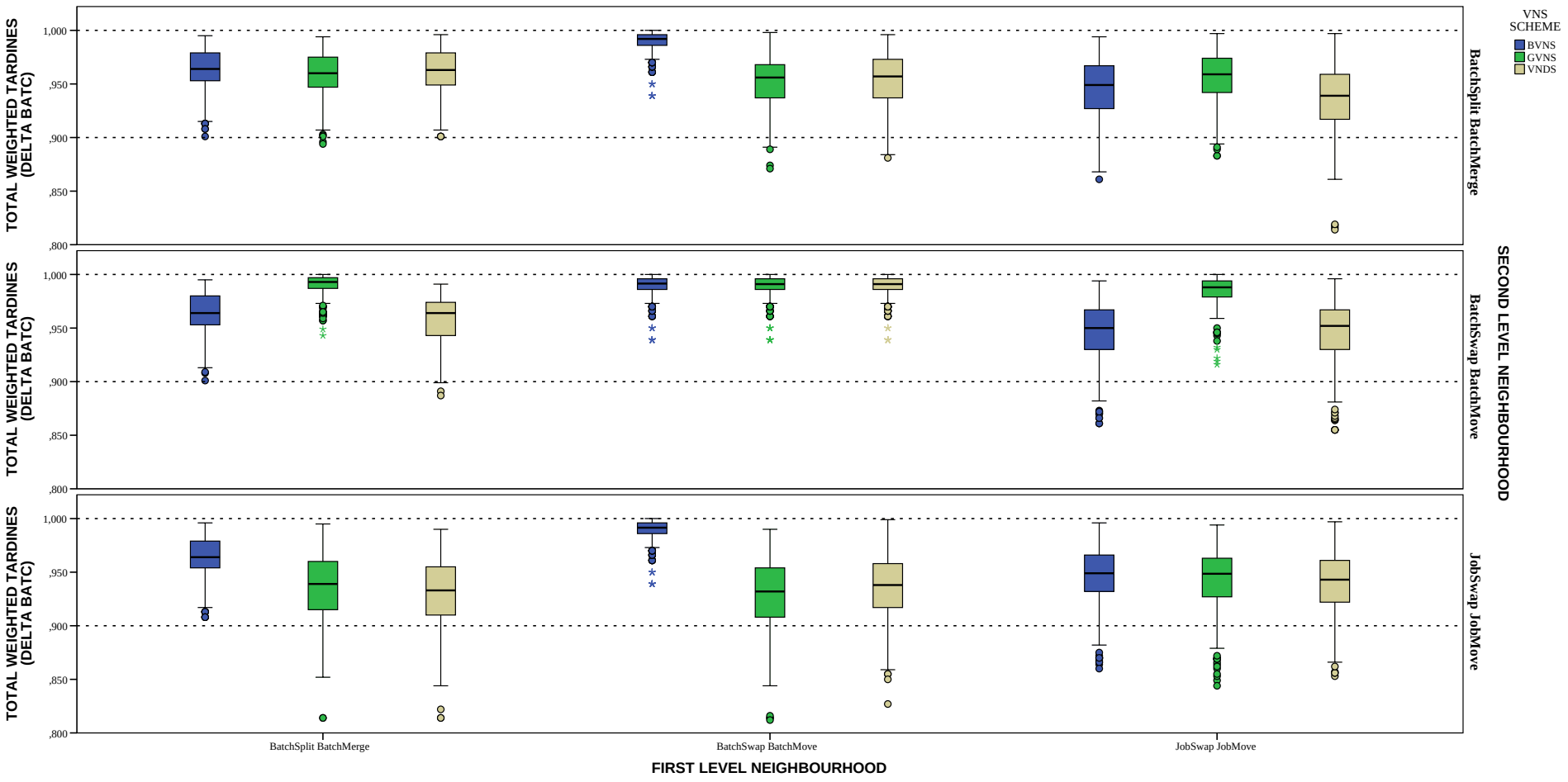
E.18 Experiment E.18

Design of experiment E.18

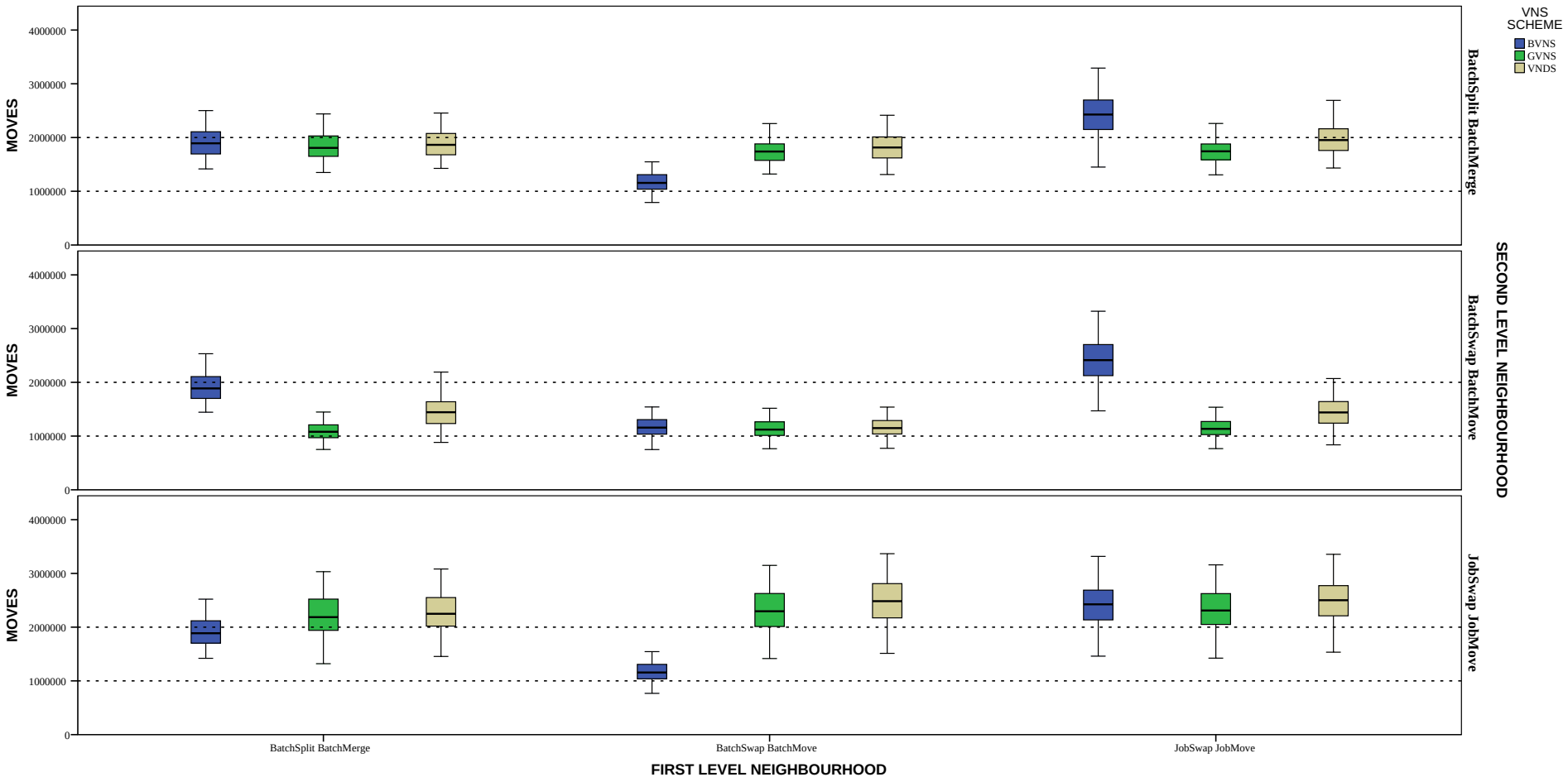
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	BVNS, GVNS, VNDS	3
deadline [min]	3	1
first level neighborhoods	Batch-Split+Merge, Batch-Swap+Move, JobSwap+Move	3
second level neighborhoods	Batch-Split+Merge, Batch-Swap+Move, JobSwap+Move	3
number of methods and settings		27
replications per run		5
total number of runs		12150



Experiment E18: total weighted tardiness improvements depending on the neighbourhood structure (Benchmark)



Experiment E18: total weighted tardiness improvements depending on the neighbourhood structure (BATC)

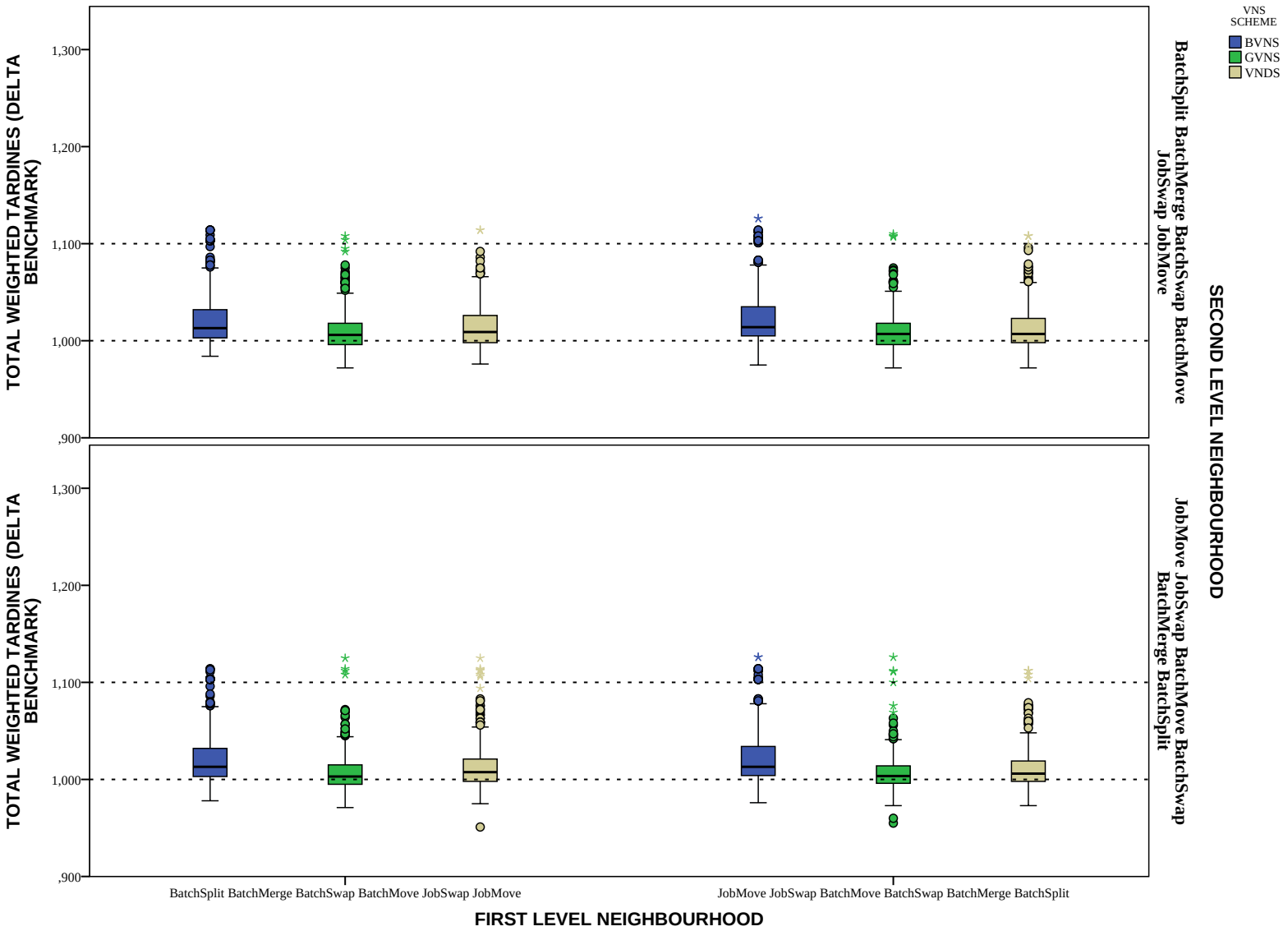


Experiment E18: moves required for total weighted tardiness improvements depending on the neighbourhood structure

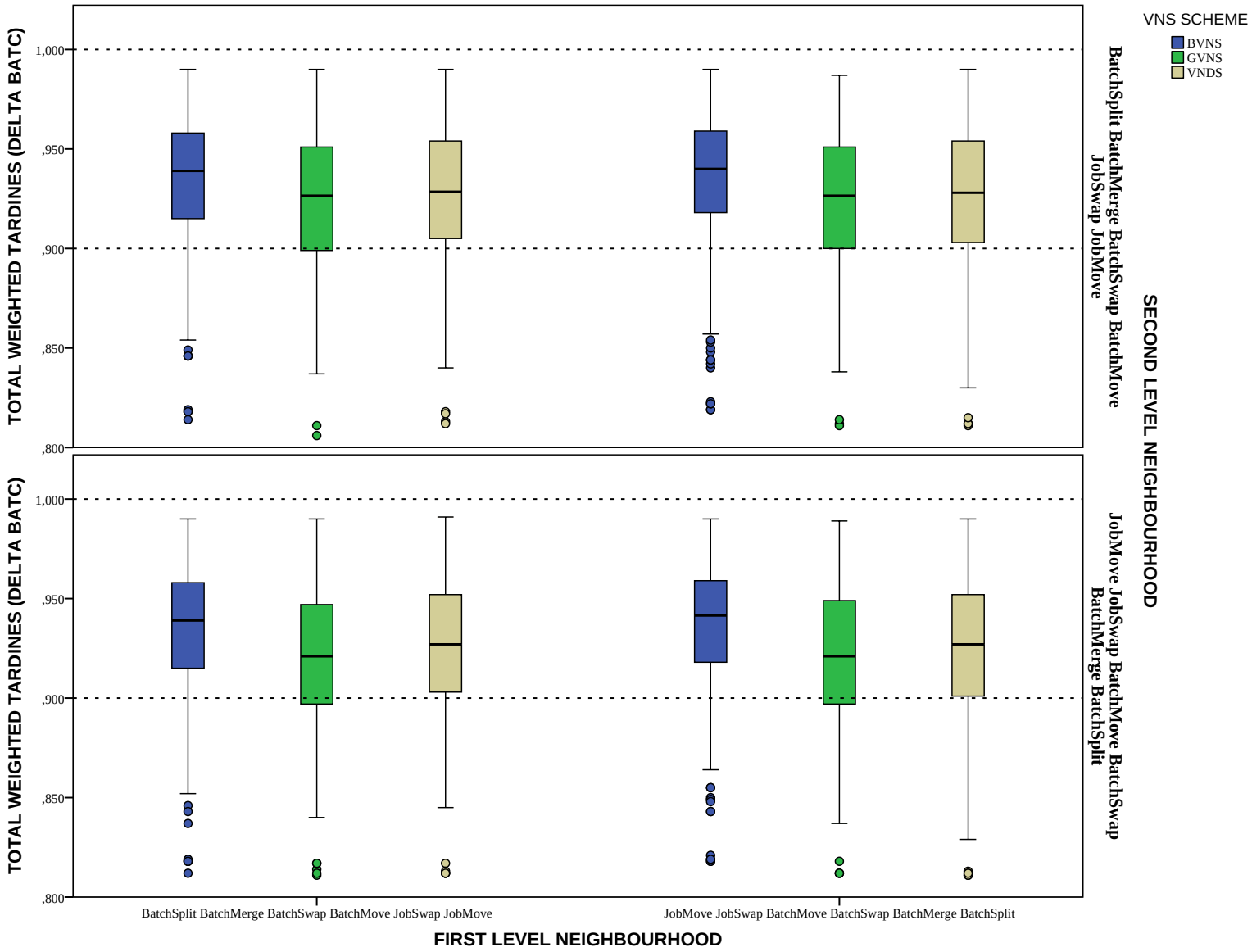
E.19 Experiment E.19

Design of experiment E.19

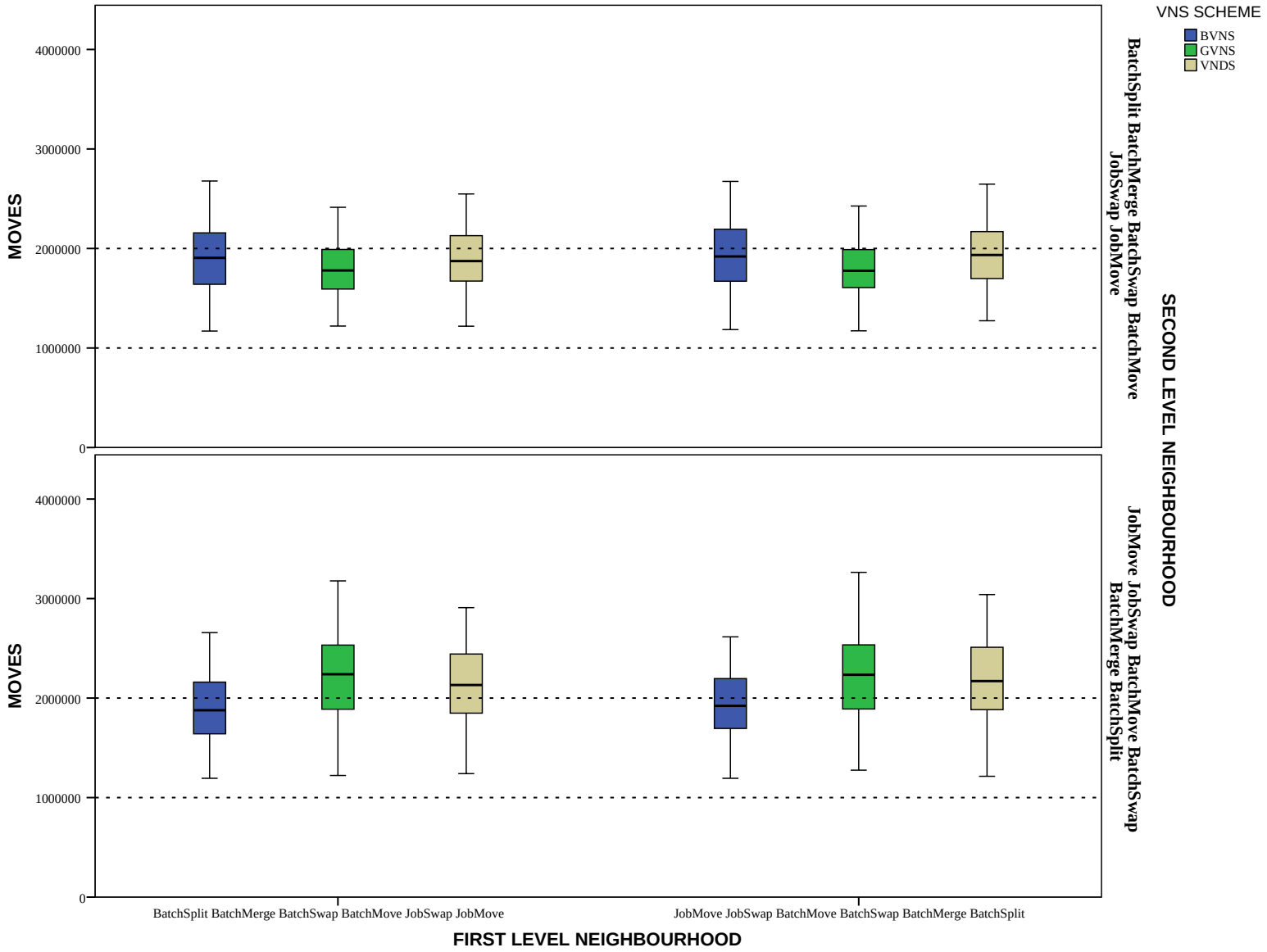
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	BVNS, GVNS, VNDS	3
deadline [min]	3	1
first level neighborhoods	default order, reverse order	2
second level neighborhoods	default order, reverse order	2
number of methods and settings		12
replications per run		5
total number of runs		5400



Experiment E19: total weighted tardiness improvements depending on the neighbourhood structure (Benchmark)



Experiment E19: total weighted tardiness improvements depending on the neighbourhood structure (BATC)

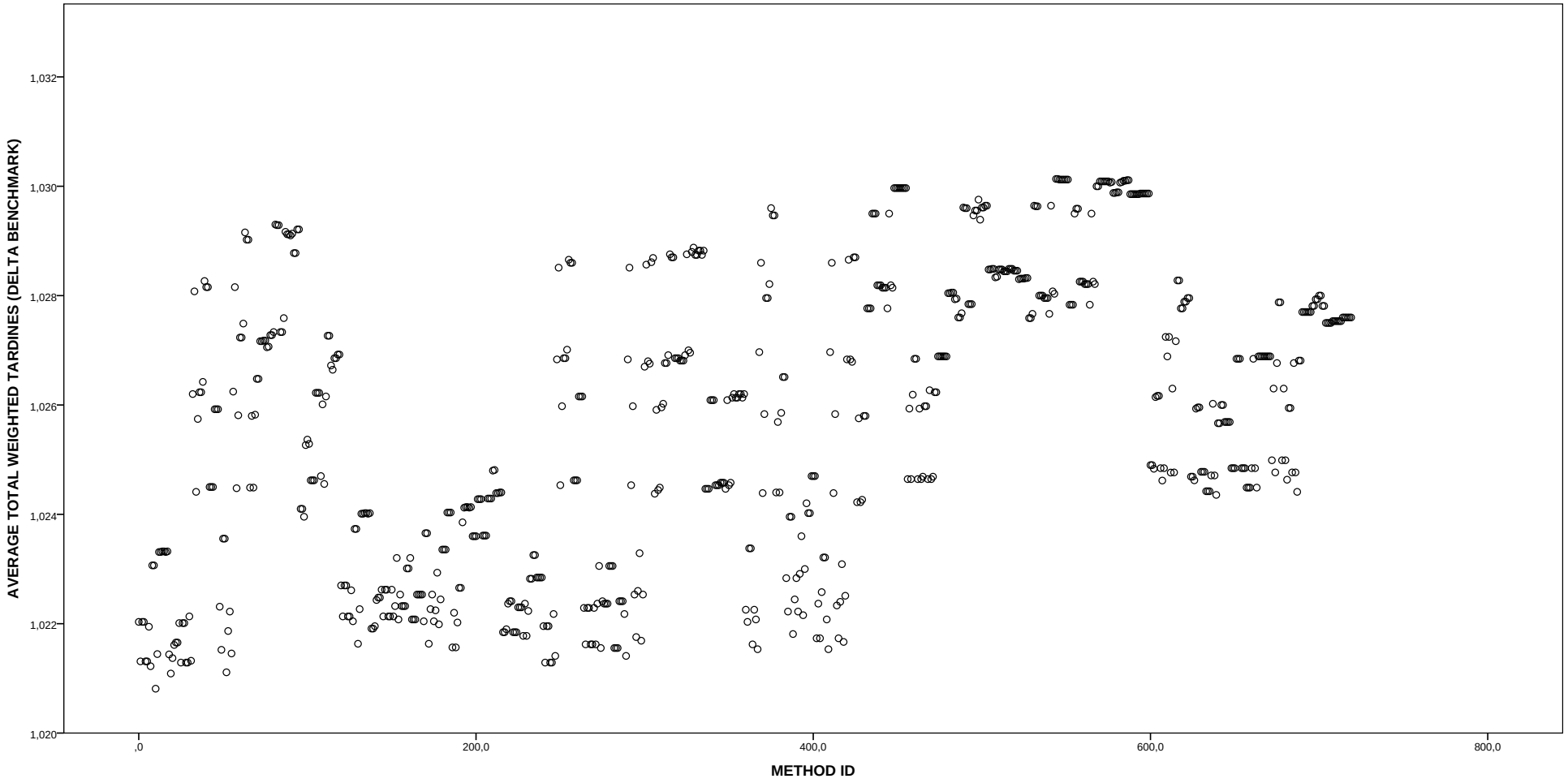


Experiment E19: moves required for total weighted tardiness improvements depending on the neighbourhood structure

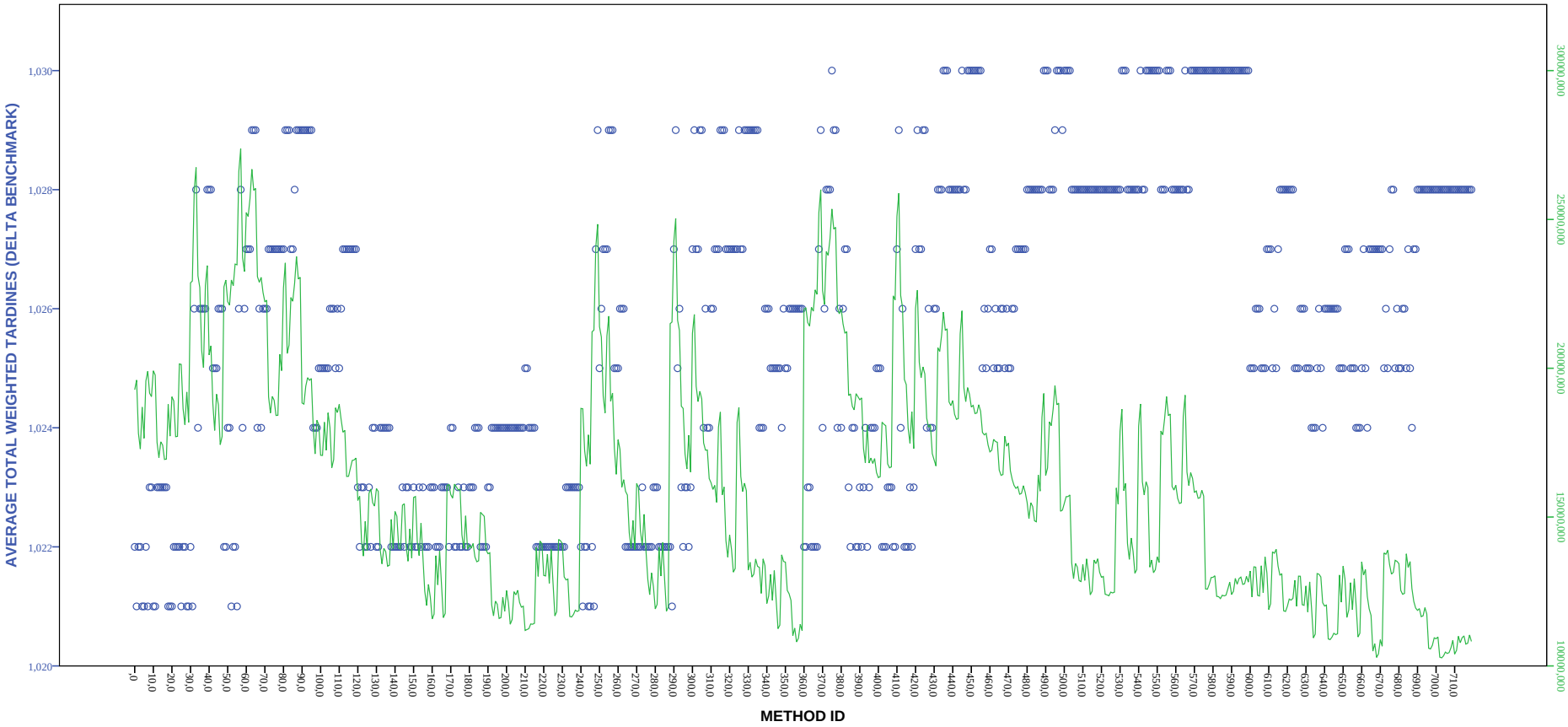
E.20 Experiment E.20

Design of experiment E.20

factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	VND	1
deadline [min]	3	1
neighborhoods	all possible sequences	720
number of methods and settings		720
replications per run		1
total number of runs		64800



Experiment E20: total weighted tardiness improvements depending on the sequence of neighbourhood (Benchmark)

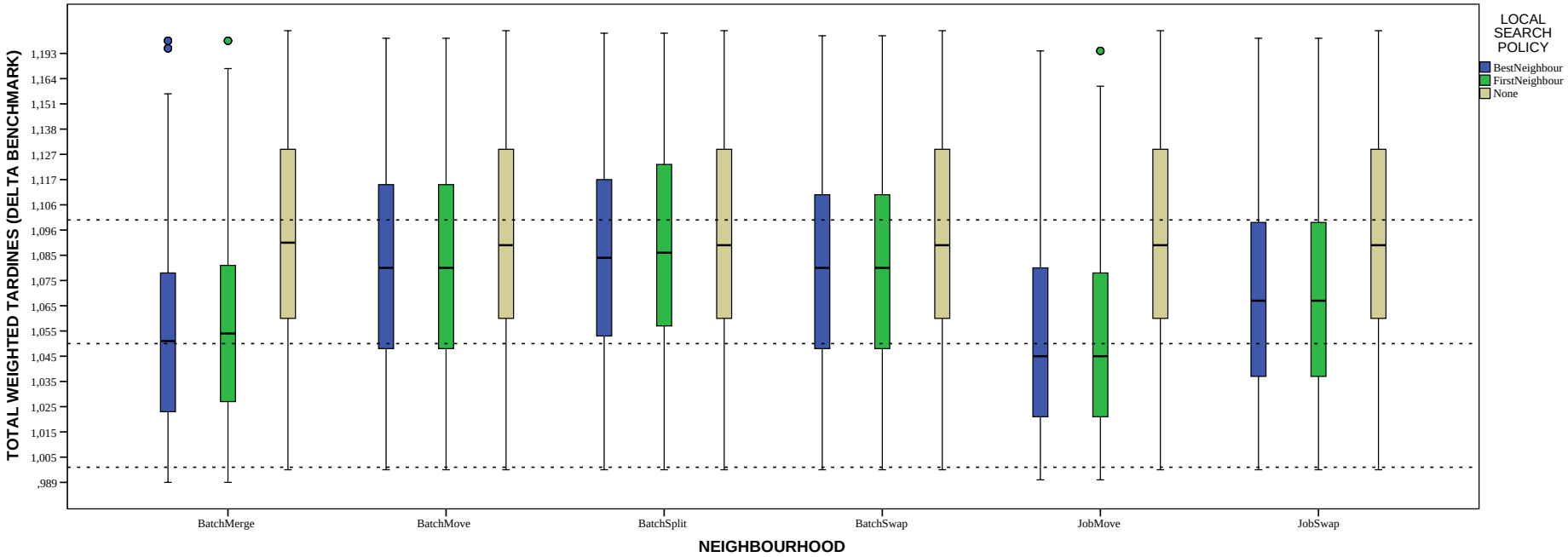


Experiment E20: total weighted tardiness improvements and required moves depending on the sequence of neighbourhood (Benchmark)

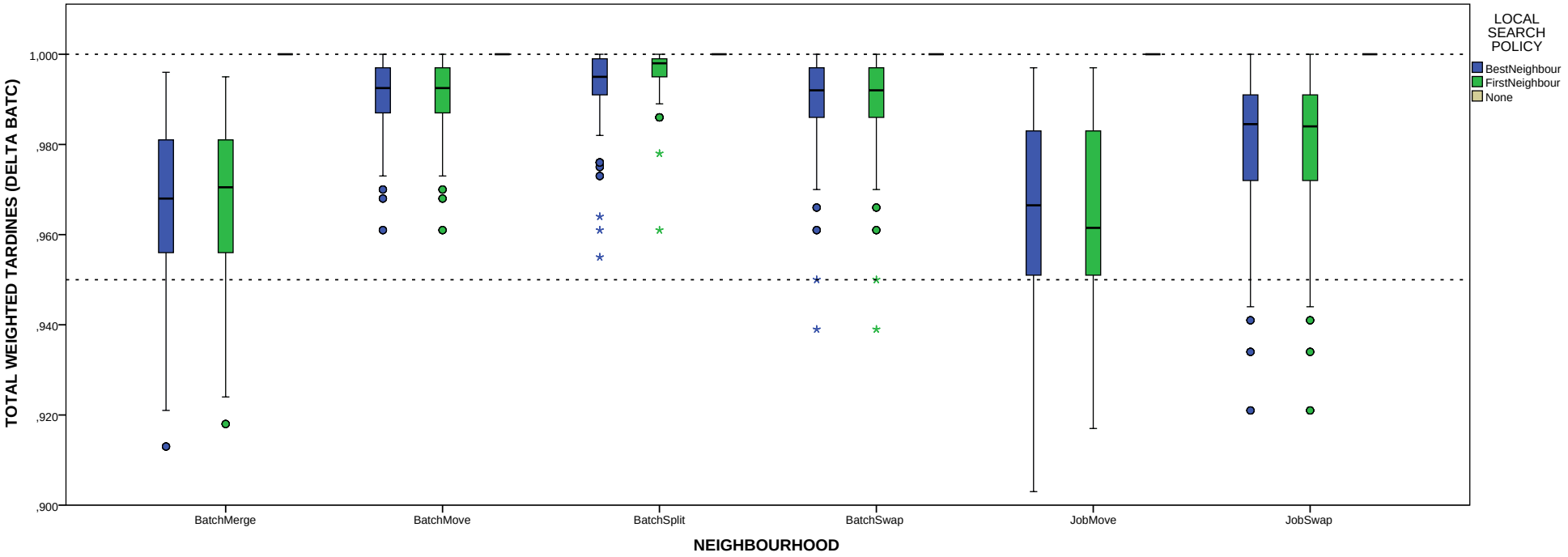
E.21 Experiment E.21

Design of experiment E.21

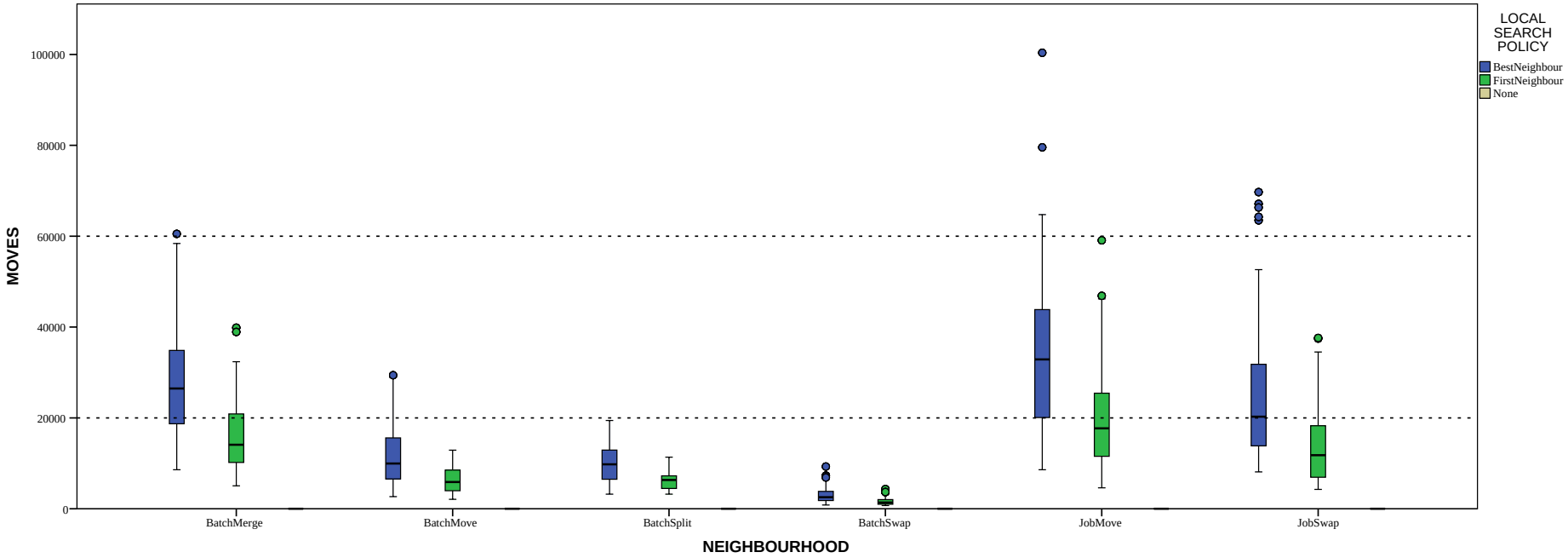
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	VND	1
deadline [min]	3	
first level neighborhoods	default neighborhood sequence	6
first level LS	none, first neighbor, best neighbor	3
number of methods and settings		18
replications per run		1
total number of runs		1620



Experiment E21: total weighted tardiness improvements depending on the local search scheme (Benchmark)



Experiment E21: total weighted tardiness improvements depending on the local search scheme (BATC)

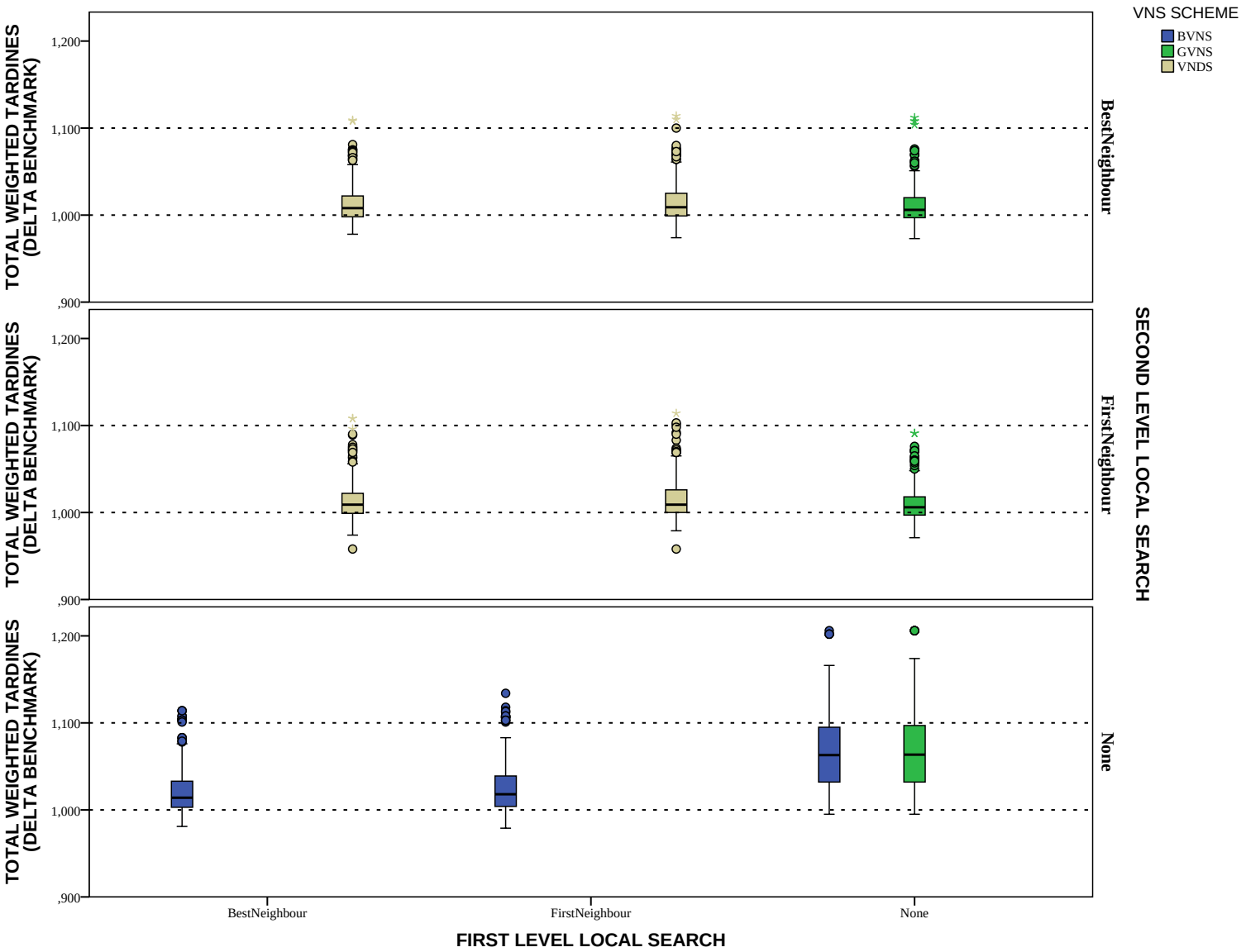


Experiment E21: moves required for total weighted tardiness improvements depending on the local search scheme

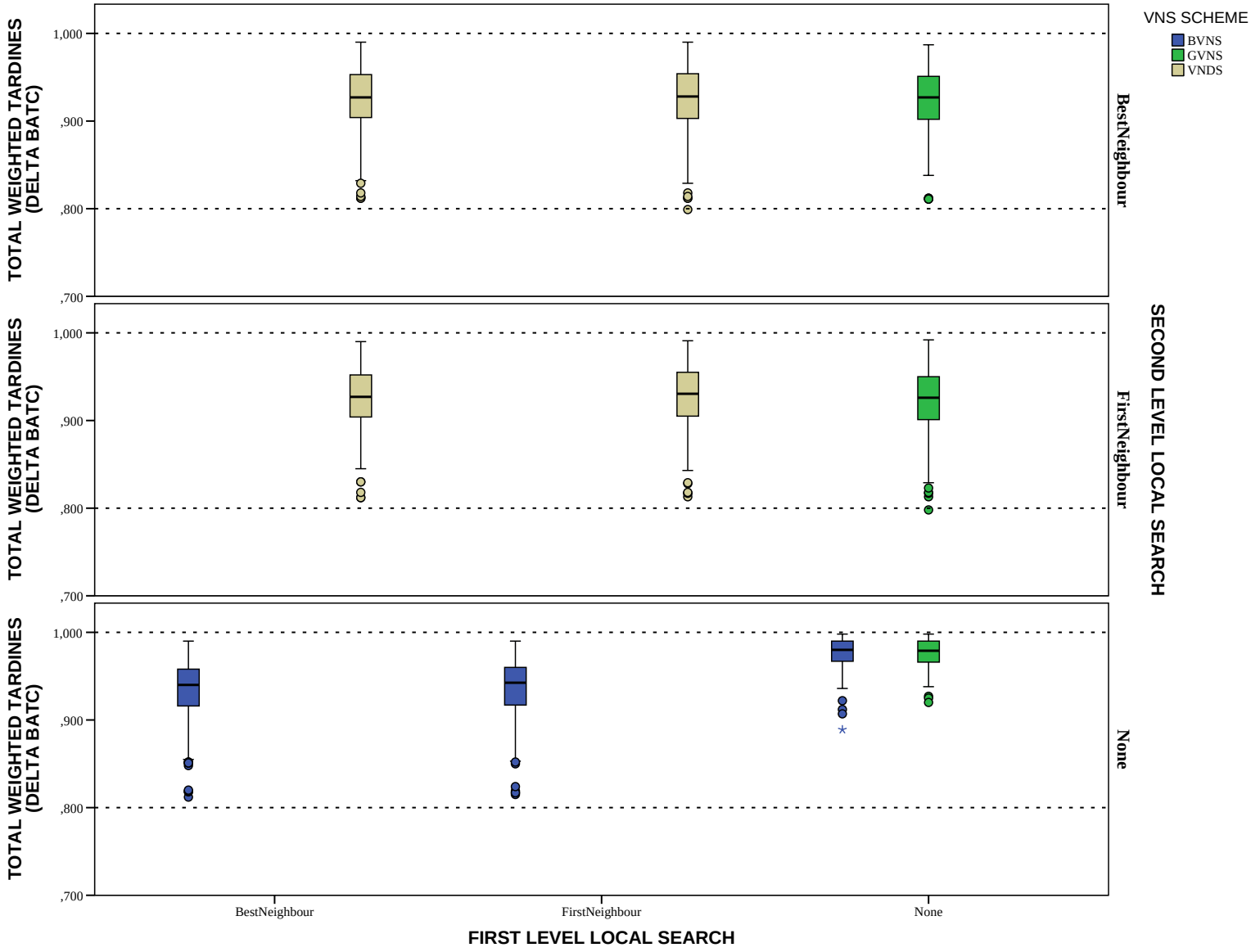
E.22 Experiment E.22

Design of experiment E.22

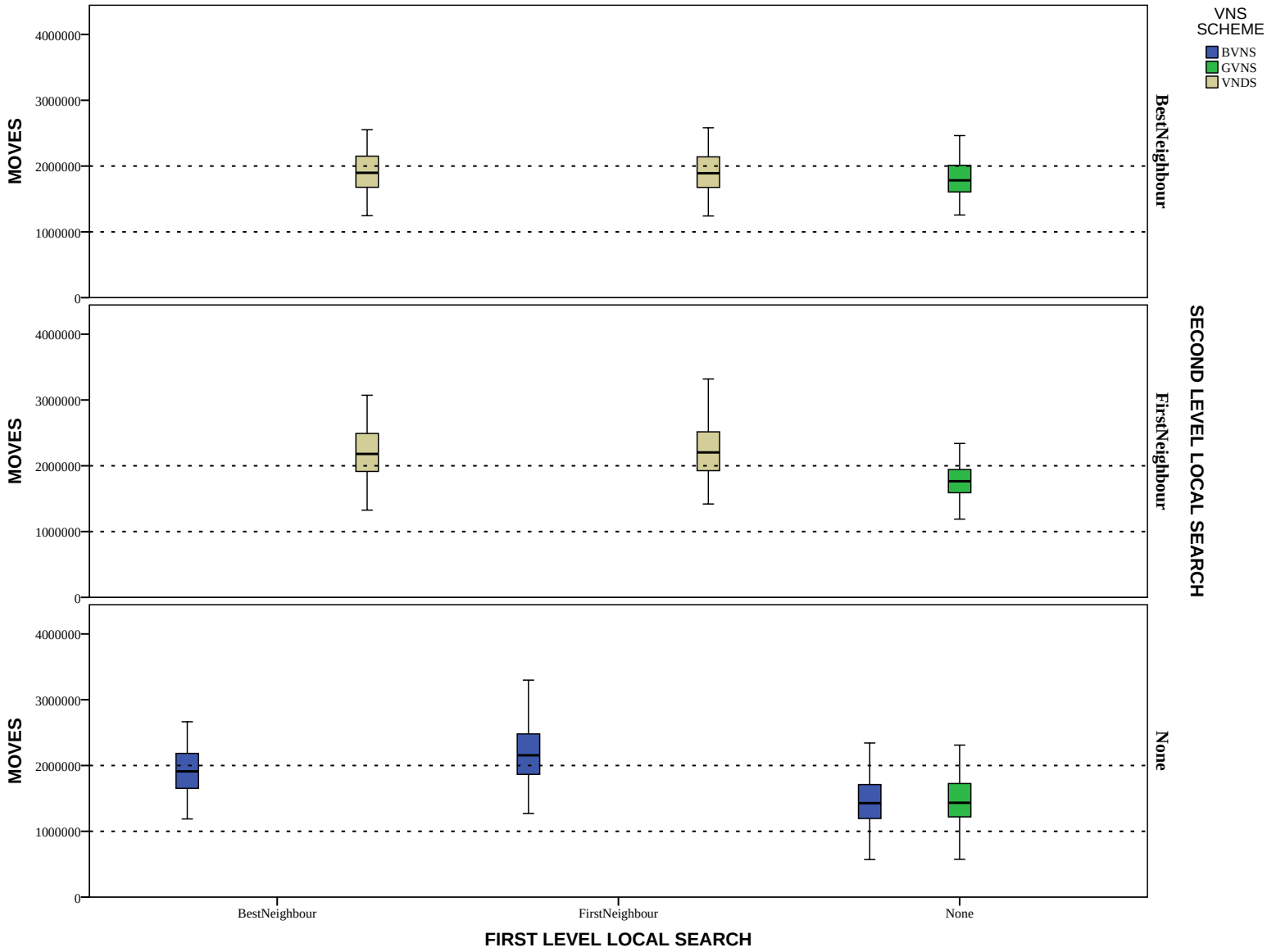
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	BVNS, GVNS, VNDS	3
deadline [min]	3	1
first level neighborhoods	default neighborhood sequence	1
first level LS	none, first neighbor, best neighbor	3
second level neighborhoods	default neighborhood sequence	1
second level LS	none, first neighbor, best neighbor	3
number of methods and settings		10
replications per run		5
total number of runs		4500



Experiment E22: total weighted tardiness improvements depending on the local search scheme (Benchmark)



Experiment E22: total weighted tardiness improvements depending on the local search scheme (BATC)

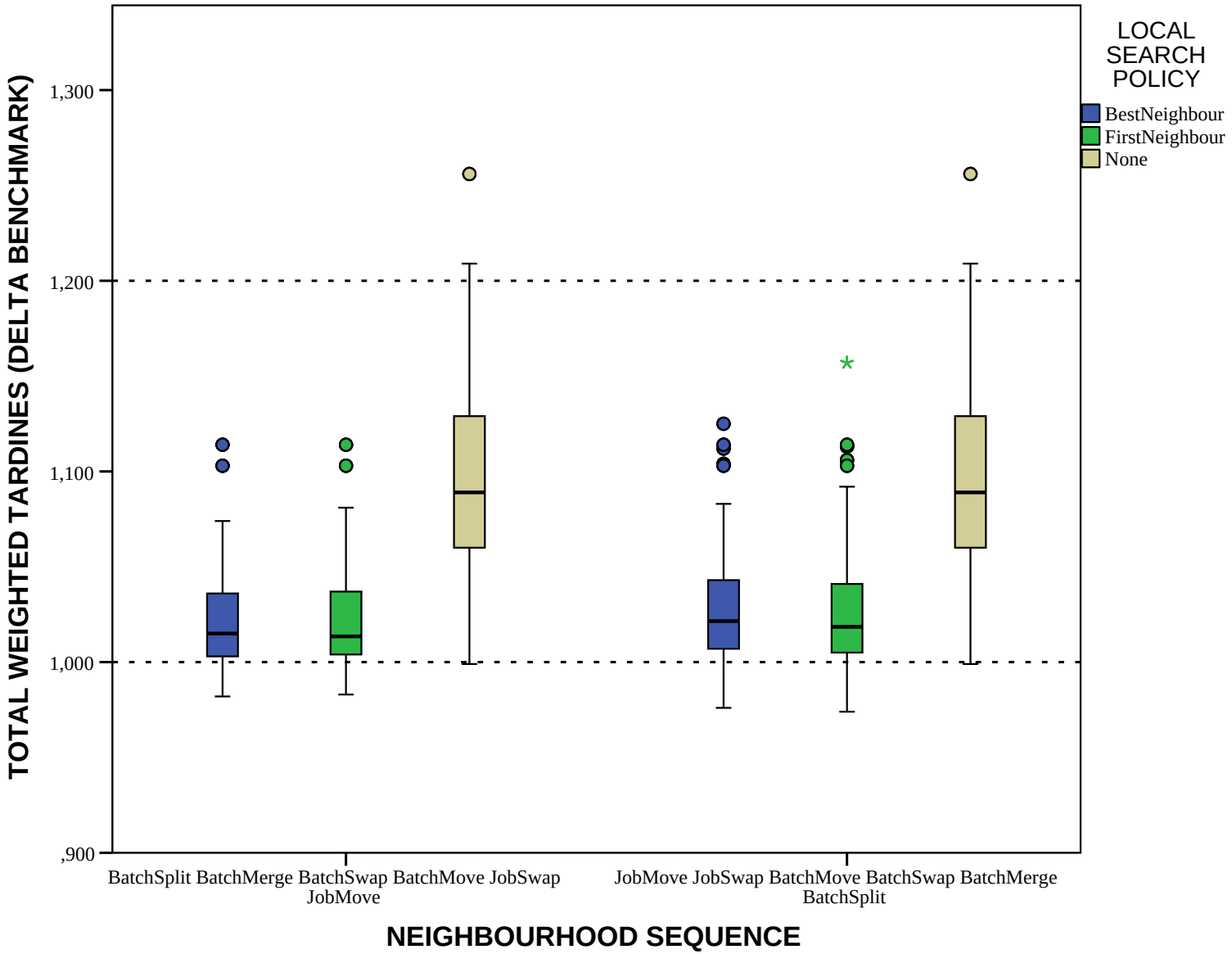


Experiment E22: moves required for total weighted tardiness improvements depending on the local search scheme

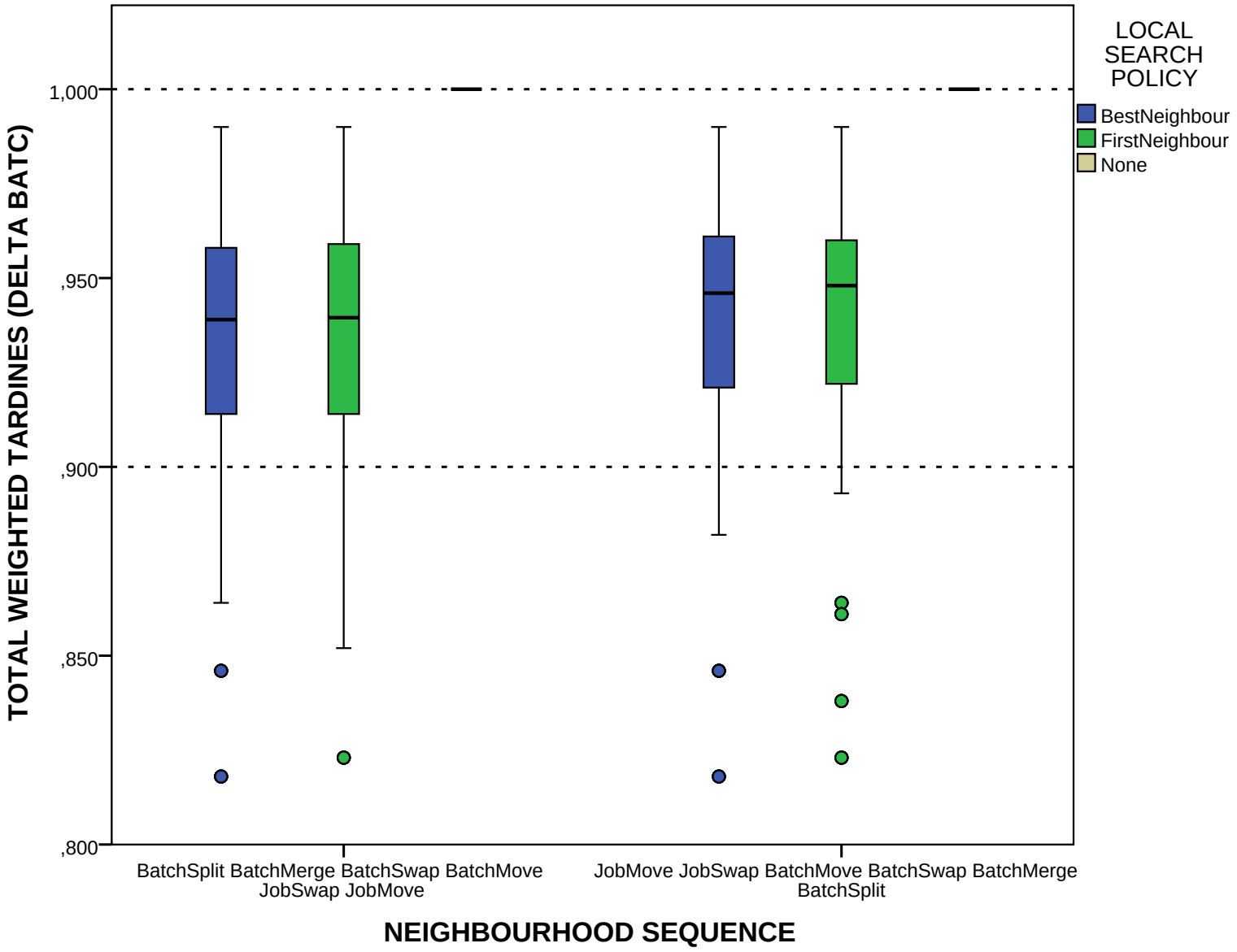
E.23 Experiment E.23

Design of experiment E.23

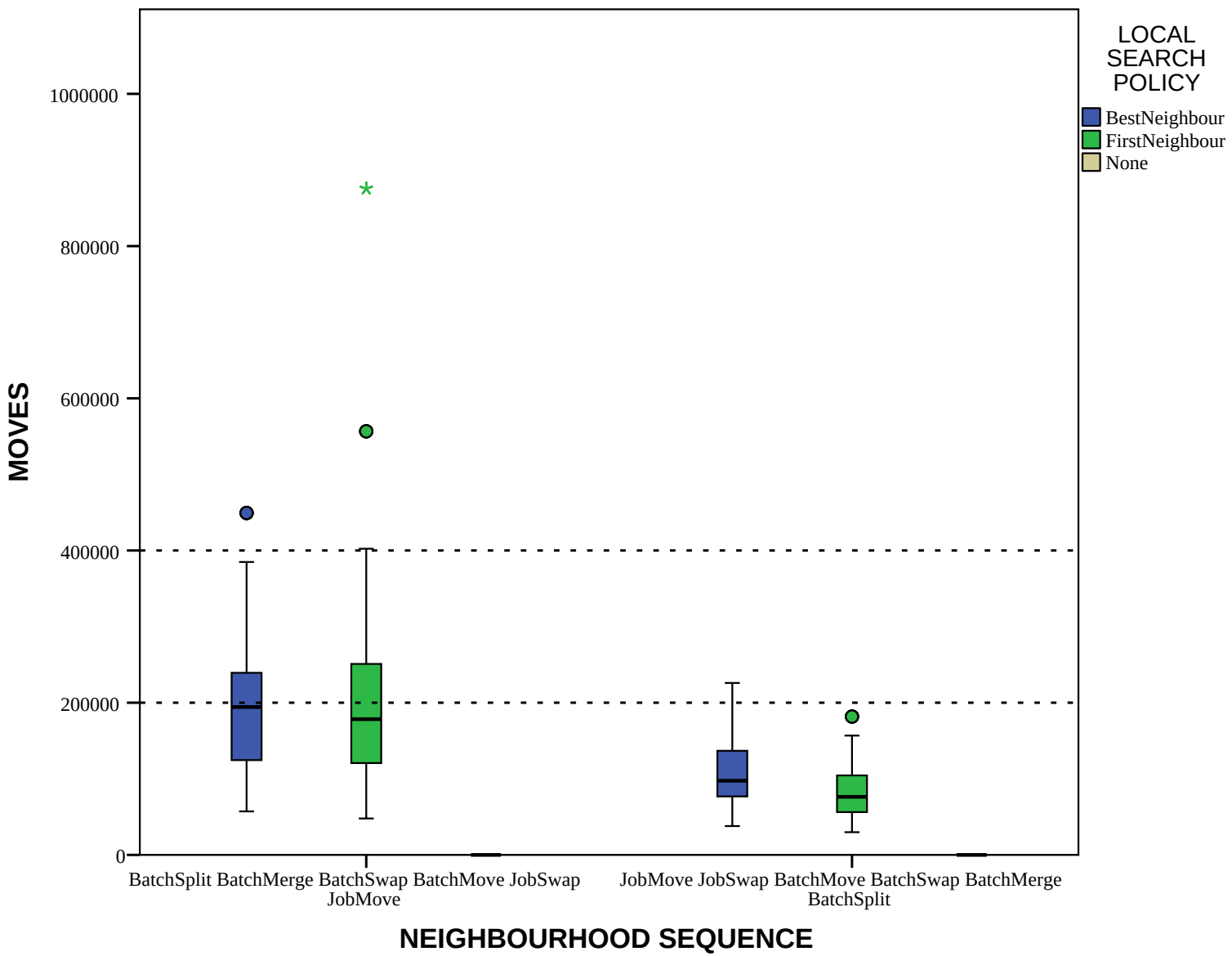
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	VND	1
deadline [min]	3	
first level neighborhoods	default order, reverse order	2
first level LS	none, first neighbor, best neighbor	3
number of methods and settings		6
replications per run		5
total number of runs		2700



Experiment E23: total weighted tardiness improvements depending on the local search scheme (Benchmark)



Experiment E23: total weighted tardiness improvements depending on the local search scheme (BATC)

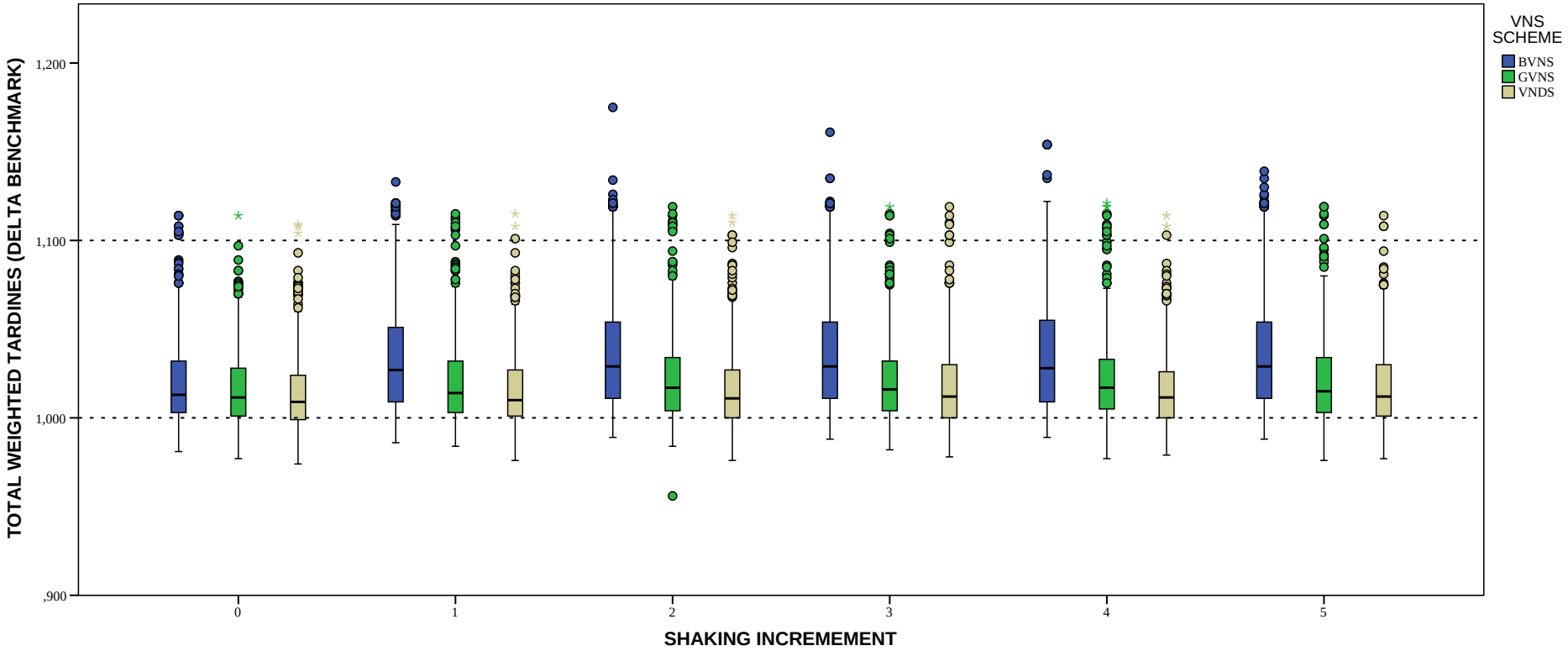


Experiment E23: moves required for total weighted tardiness improvements depending on the local search scheme

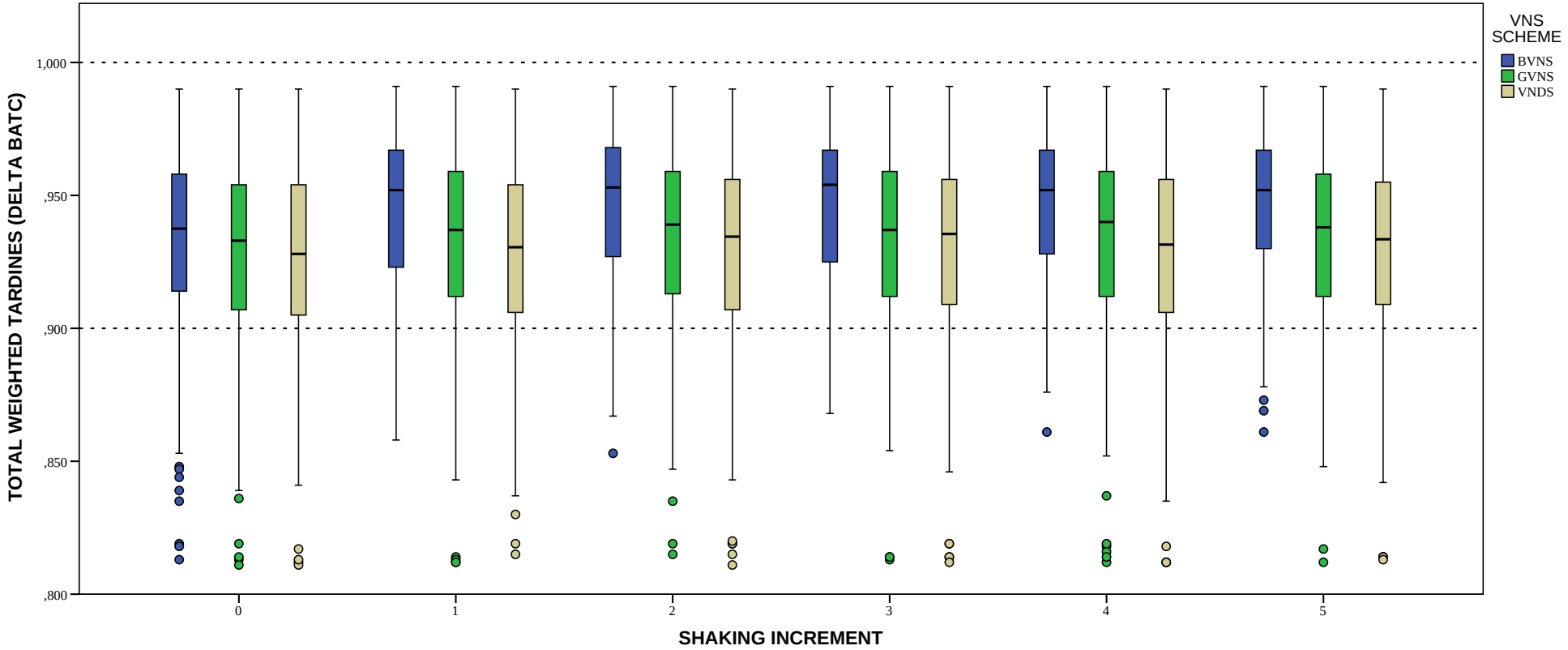
E.24 Experiment E.24

Design of Experiment E.24

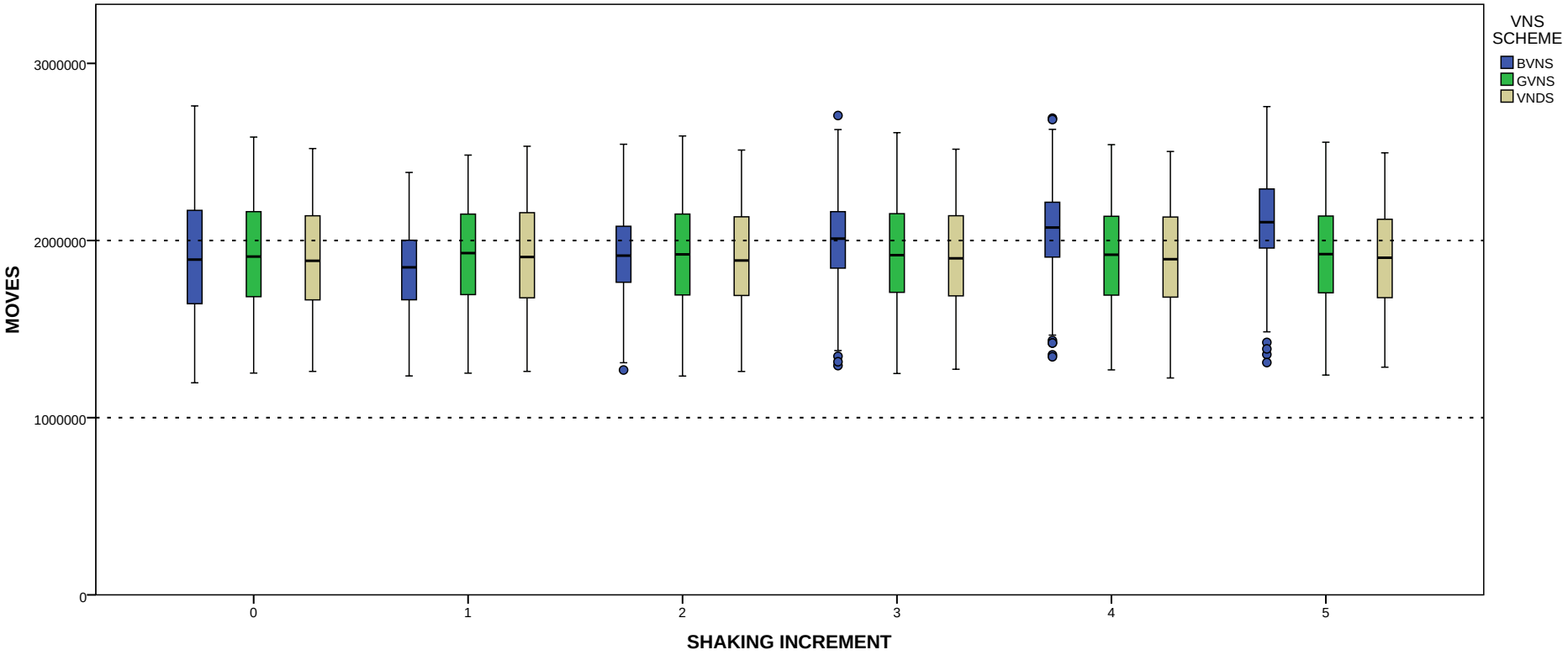
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	BVNS, GVNS, VNDS	3
deadline [min]	3	
neighborhoods	default neighborhood sequence	1
shaking steps	0, 1, 2, 3, 4, 5	6
number of methods and settings		18
replications per run		5
total number of runs		8100



Experiment E24: total weighted tardiness improvements depending on the shaking range (Benchmark)



Experiment E24: total weighted tardiness improvements depending on the shaking range (BATC)

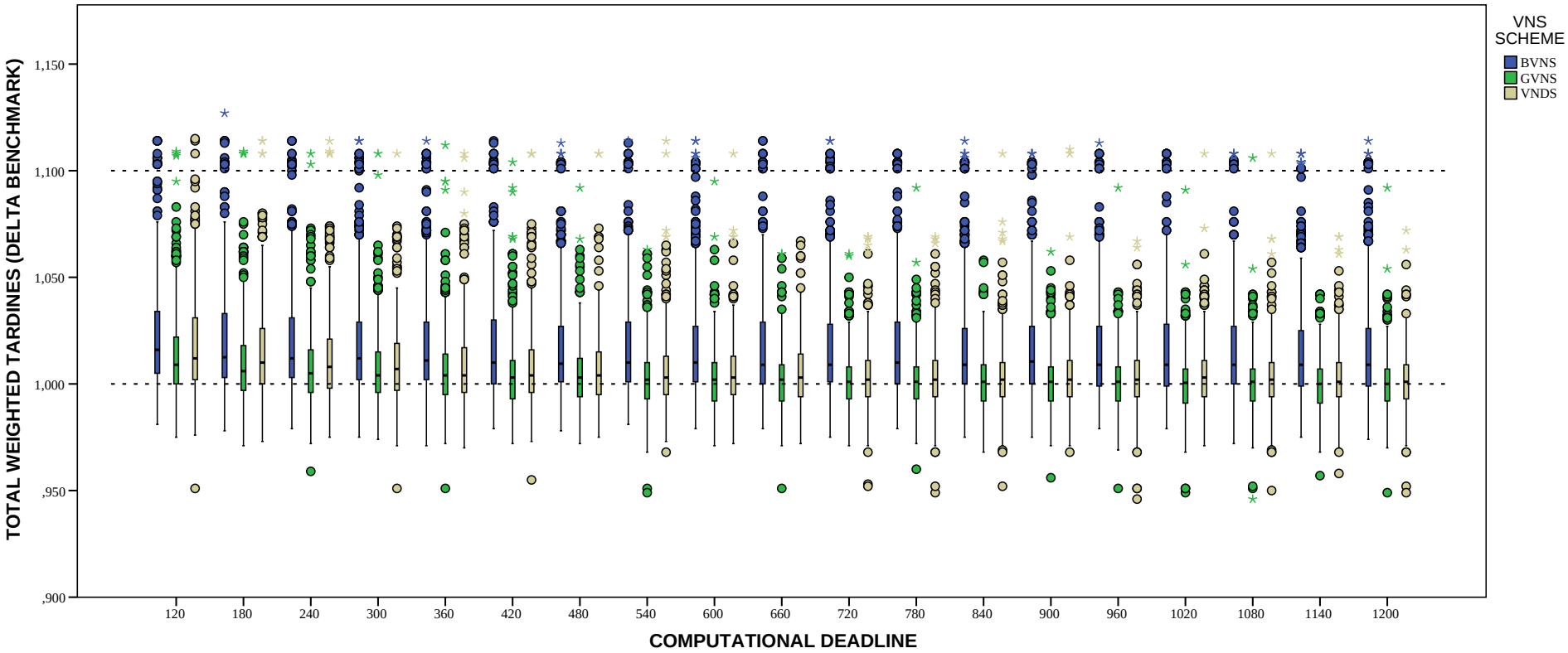


Experiment E24: moves required for total weighted tardiness improvements depending on the shaking range

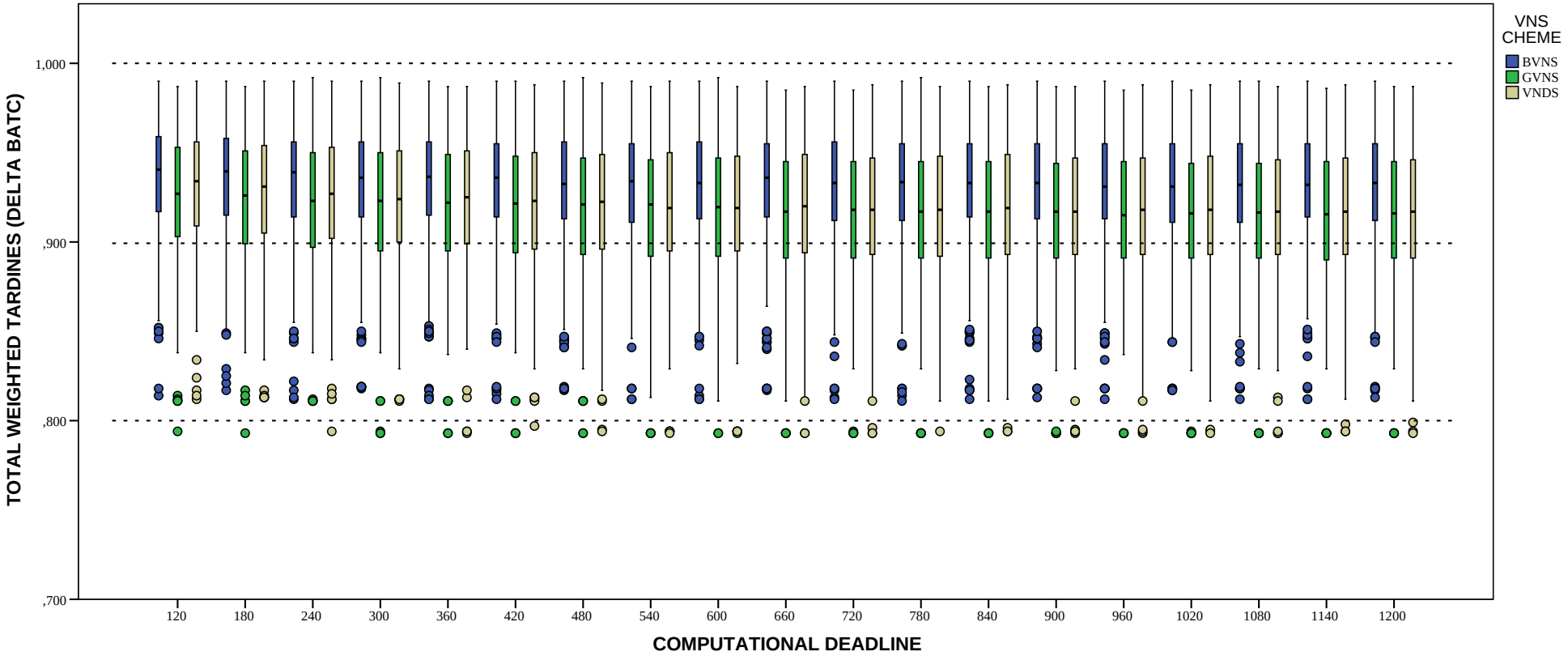
E.25 Experiment E.25

Design of experiment E.25

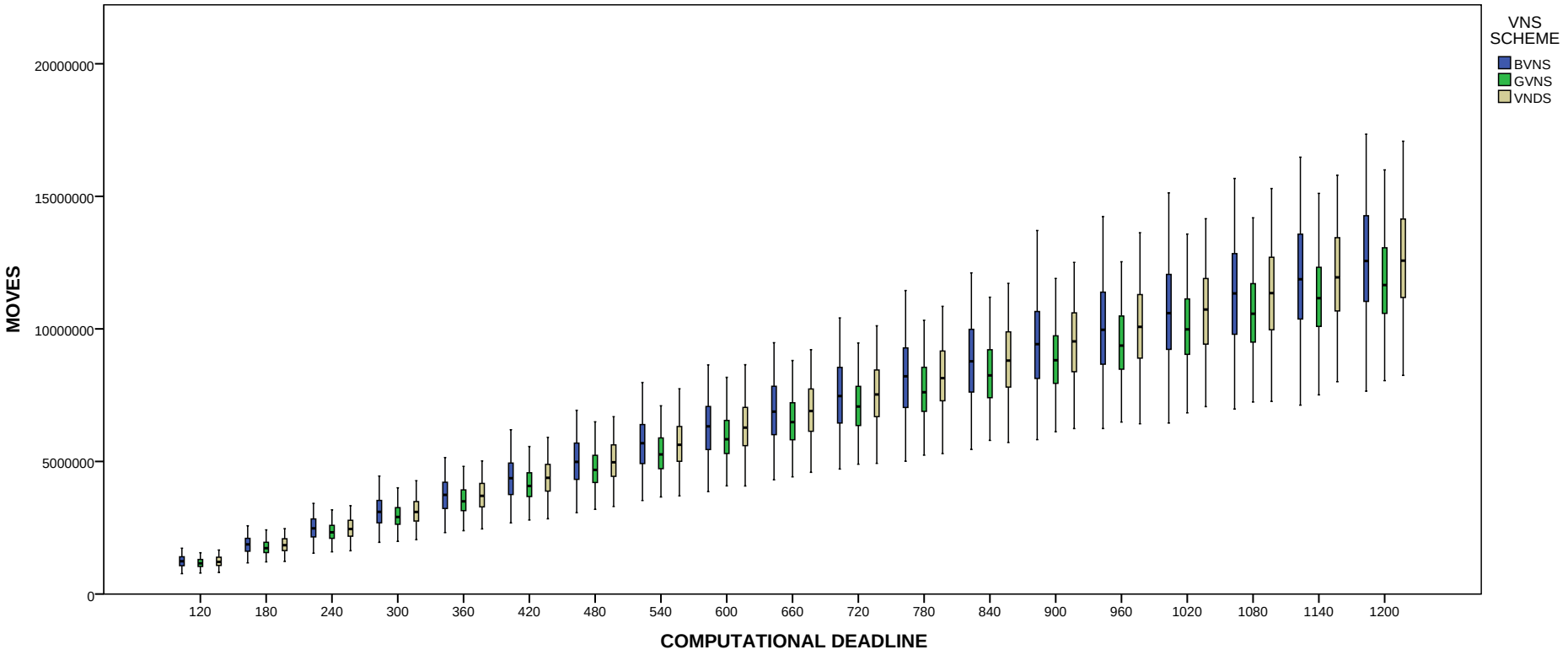
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	BVNS, GVNS, VNDS	3
deadline [min]	2,3,4, . . . , 20	19
neighborhoods	default neighborhood sequence	1
shaking steps	1	1
number of methods and settings		57
replications per run		5
total number of runs		25650



Experiment E25: total weighted tardiness improvements depending on the computational deadline (Benchmark)



Experiment E25: total weighted tardiness improvements depending on the computational deadline (BATC)

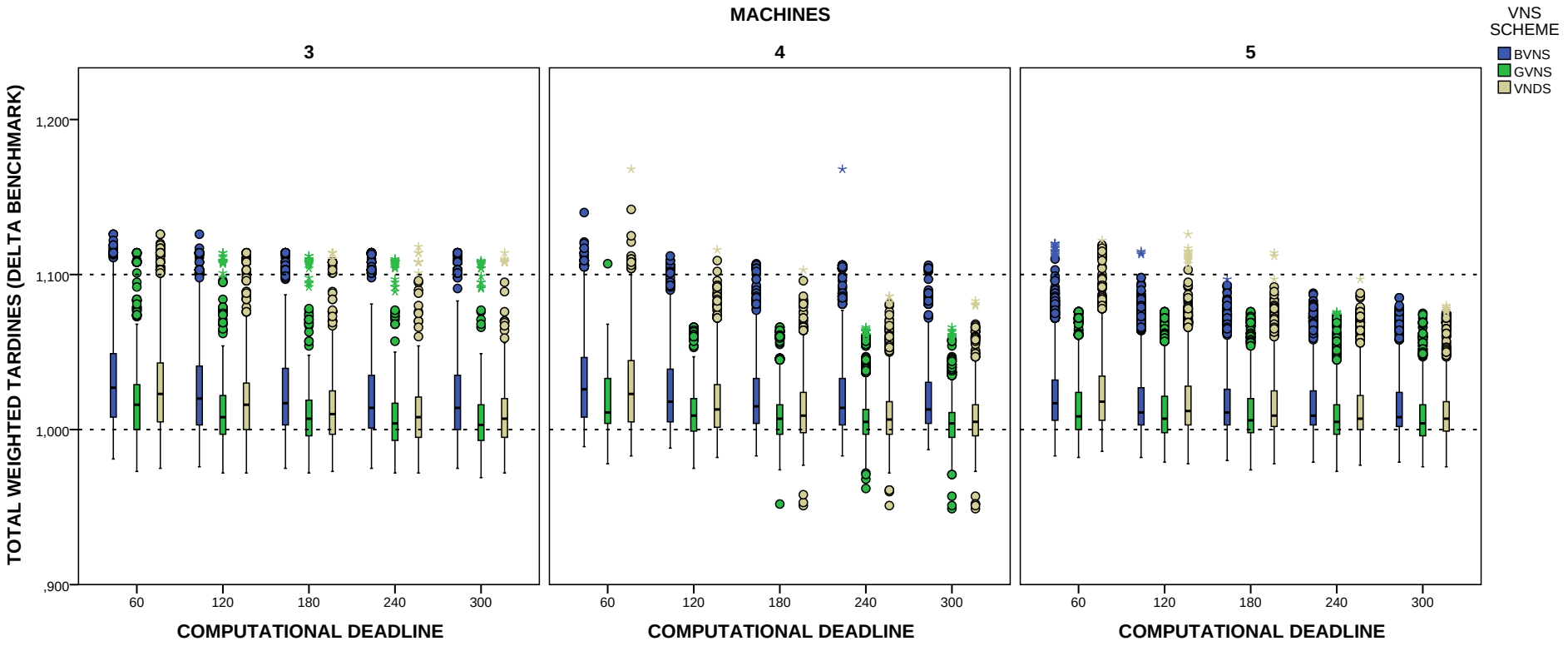


Experiment E25: moves required for total weighted tardiness improvements depending on the computational deadline

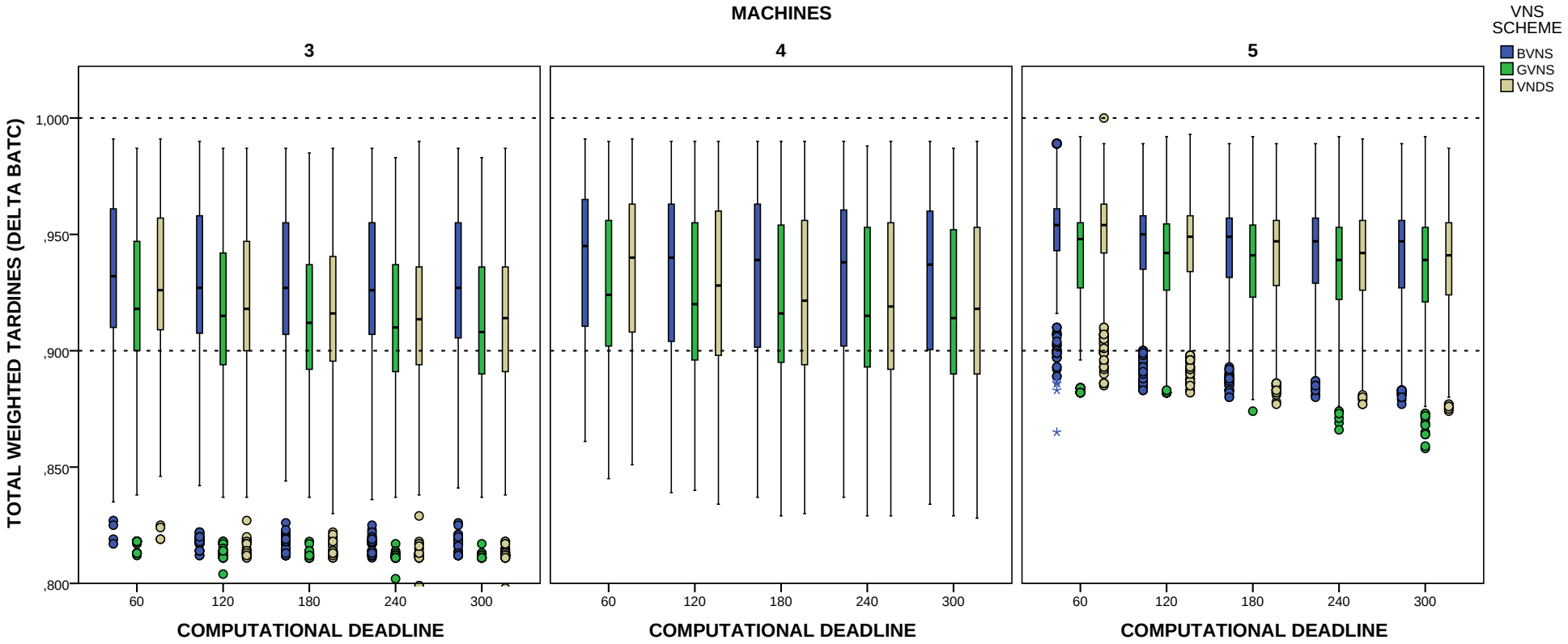
E.26 Experiment E.26

Design of experiment E.26

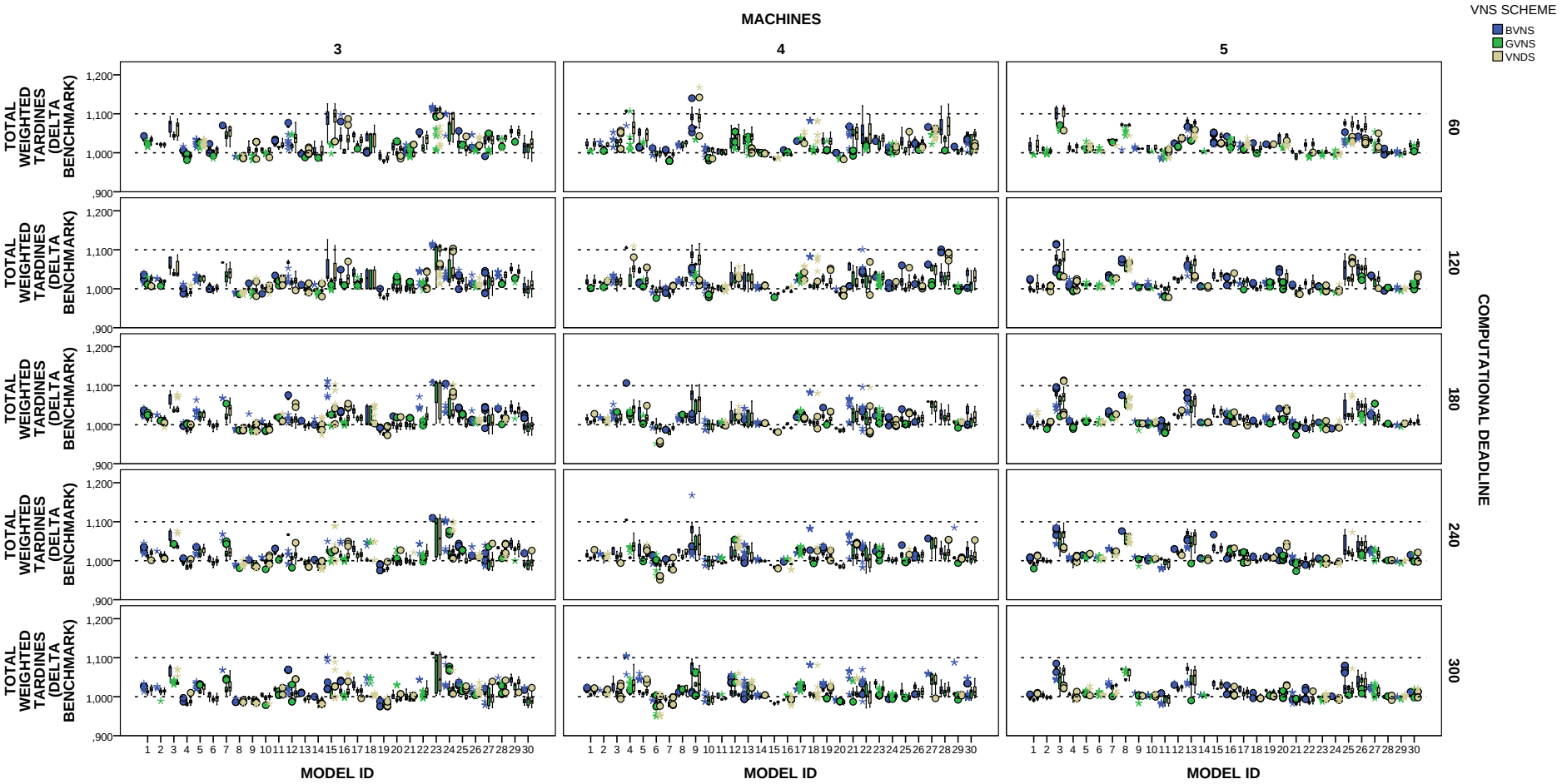
factor	level	count
total number of model instances		90
initial solution	BATC	1
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	BVNS, GVNS, VNDS	3
deadline [min]	1,2,3,4,5	5
neighborhoods	default neighborhood sequence	1
shaking steps	1	1
number of methods and settings		15
replications per run		50
total number of runs		67500



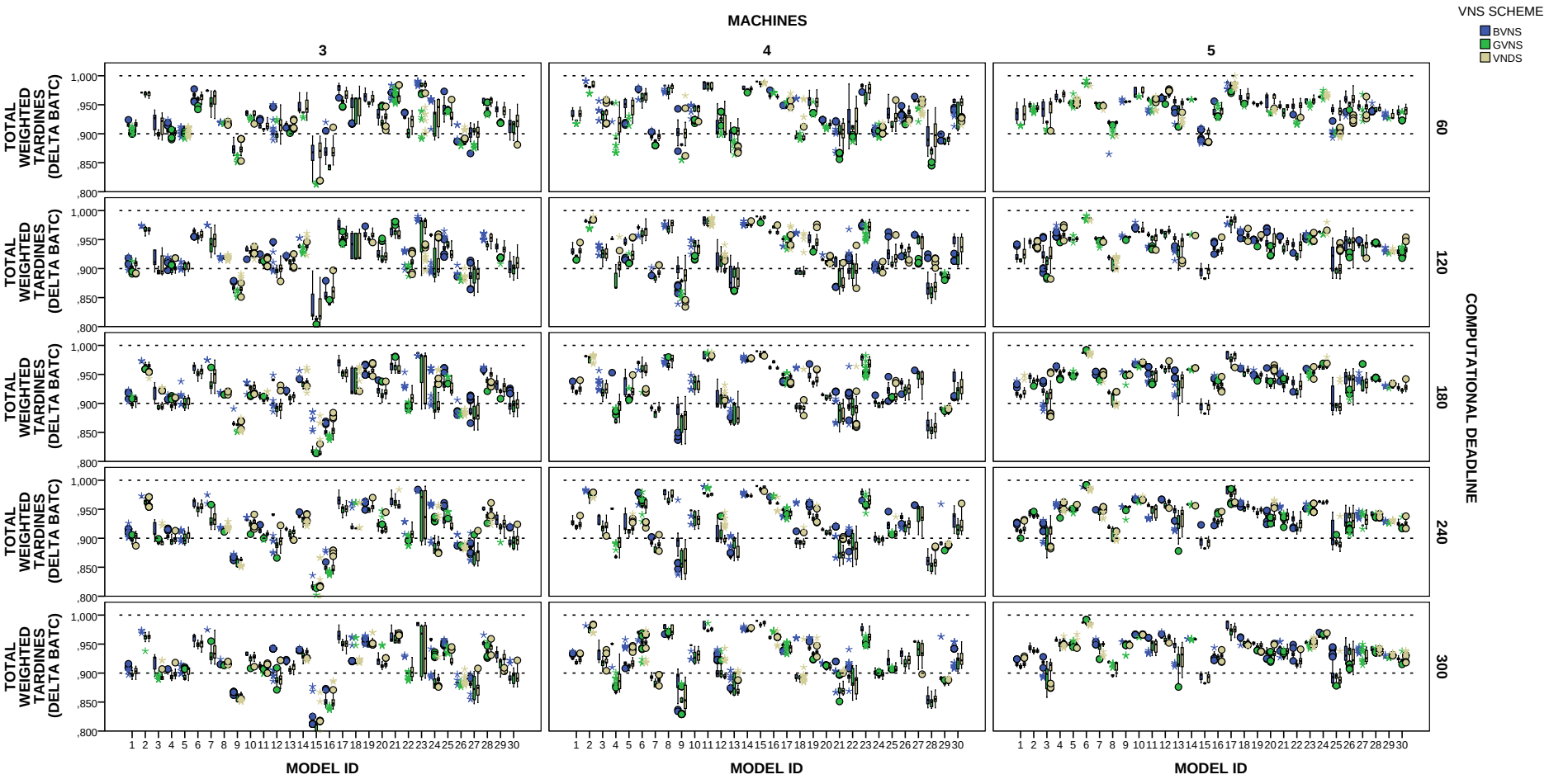
Experiment E26: total weighted tardiness improvements depending on the computational deadline (Benchmark)



Experiment E26: total weighted tardiness improvements depending on the computational deadline (BATC)



Experiment E26: total weighted tardiness improvements depending on the computational deadline (Benchmark)

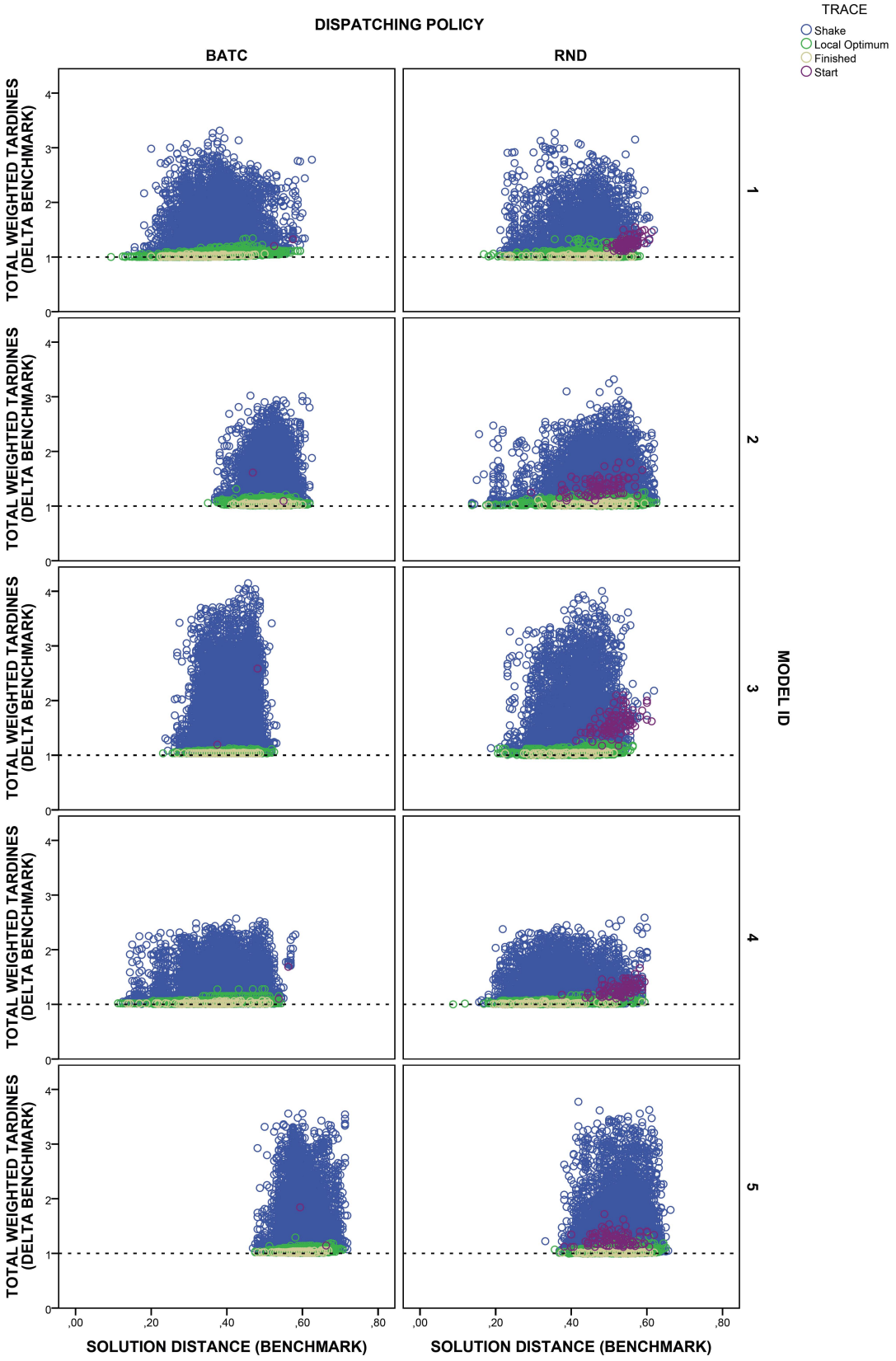


Experiment E26: total weighted tardiness improvements depending on the computational deadline (BATC)

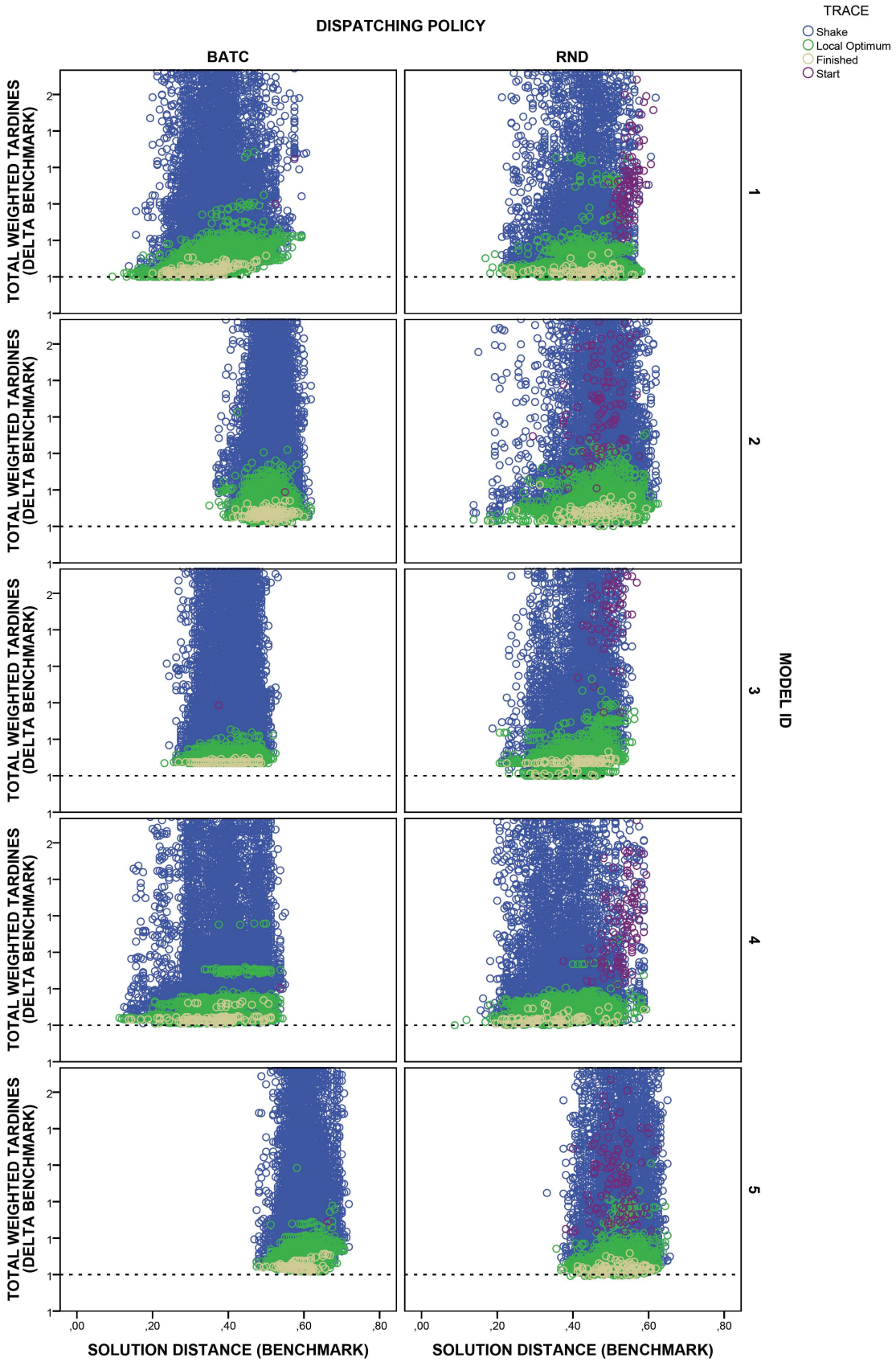
E.27 Experiment E.27

Design of experiment E.27

factor	level	count
total number of model instances		90
initial solution	random (RND), BATC	2
BATC parameter	best of (0.3, 0.4, 0.5, 0.6, 0.7)	1
reference solution	initial solution	1
objective(s)	TWT	1
VNS type	GVNS	1
deadline [min]	10	1
neighborhoods	default neighborhood sequence	1
shaking steps	1	1
number of methods and settings		2
replications per run		100
total number of runs		18000



Experiment E27: visualization of search traces (view1)



Experiment E27: visualization of search traces (view2)