Universität der Bundeswehr München
Fakultät für Informatik
Institut für Technische Informatik

Algorithmengesteuerte Modulare Produktion:
zentrales, dezentrales und lernendes Scheduling

Sebastian Mayer

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 der Dissertation:

1. Prof. Dr.-Ing. Christian Endisch
2. Prof. Dr. rer. nat. Oliver Rose

Die Dissertation wurde am 02.08.2022 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 04.11.2022 angenommen. Die mündliche Prüfung fand am 19.12.2022 statt.

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Innovative Mobilität der Technischen Hochschule Ingolstadt im Rahmen eines Kooperationsprojektes mit der Audi AG. Mein besonderer Dank gilt dem Institutsleiter Prof. Dr.-Ing. Christian Endisch. Christian, danke für dein Vertrauen, die wissenschaftlichen Freiheiten, die ich genießen durfte und die fortwährende Unterstützung meiner Arbeit. Zu Beginn meiner Doktorandenzeit erzähltest du mir, dass ein Promotionsprojekt ein sehr prägendes und lehrreiches Vorhaben darstelle. Danke, dass du mich mit dieser Aussage motiviert hast, diesen Weg einzuschlagen.

Für das Interesse an der Arbeit und die Übernahme des Koreferats spreche ich Herrn Prof. Dr. rer. nat. Oliver Rose einen herzlichen Dank aus. Der Austausch mit seinen Mitarbeitern Dr. Thomas Mayer und Dr. Tao Zhang zu Themen der Modellbildung und Simulation unterstützte mich bei der Fertigstellung der vorliegenden Arbeit.

Weiter gebührt mein Dank Herrn Dr. Dickhaus – für das Vertrauen, die Unterstützung und die durchweg herzliche Zusammenarbeit. Ein wichtiges Dankeschön geht auch an meine Kollegen auf Hochschulseite, die mich während der Zeit begleitet haben: Michael Okon, Patrick Herrmann, Martin Gerngroß, Lukas Leitner, Markus Kohler, Martin Nachtsheim und viele mehr. Danke für das humorvolle Arbeitsklima und die gegenseitige Unterstützung. Auf Audiseite danke ich herzlichst meinen Kollegen und Kolleginnen Veronika Stübler, Marc Hermans, Jörg Schmidt, Jürgen Glaab, Mark Andresen, Wolfgang Kern, Pascal Piszczek und auch hier vielen mehr. Ob ihr mich in Arbeitsthemen, privaten Themen, bei Bandproben oder Workshops begleitet habt, ihr habt meinen Arbeitsalltag bereichert. Außerdem danke ich allen, die mich bei der Fertigstellung der Arbeit durch kritisches Feedback unterstützt haben. Felix Gehlhoff und Marvin May danke ich für den offenen, kollegialen und universitätsübergreifenden Austausch zu Scheduling-Themen.

So eine Arbeit entsteht auch nicht ohne die Unterstützung zahlreicher Studenten. Danke für euren offenen Umgang, für euren Einsatz und euer Interesse am Thema der vorliegenden Arbeit. Besonders möchte ich Dennis Gankin und Tobias Claßen danken, mit denen aus der Zusammenarbeit Freundschaften entstanden.

Zu guter Letzt geht mein Dankeschön an meine Familie, dafür, dass ihr immer für mich da seid. Danke auch an Uschi und Charly für eure liebevolle Motivation und Unterstützung. Der wichtigste und größte Dank geht an meine Freundin Kate. Du hast mich ununterbrochen unterstützt, mir den Rücken frei gehalten, persönlich Abstriche gemacht und mich durch diese Zeit getragen. Dir widme ich diese Arbeit von Herzen.

München, im September 2021 Sebastian Mayer

Kurzfassung

Im Kontext der vierten industriellen Revolution, dem Trend der Individualisierung sowie der gesetzlich getriebenen Entwicklung neuer Antriebskonzepte entwickelte sich in der Automobilindustrie das Produktionsparadigma der Modularen Produktion mit entkoppelt angeordneten Stationen und einem flexiblen Materialtransport durch ein Fahrerloses Transportsystem. Zur Nutzung der Effizienzvorteile der Modularen Produktion herrscht Handlungsbedarf in der Entwicklung effektiver Steuerungssysteme. Die vorliegende Arbeit adressiert diesen Handlungsbedarf mit der Entwicklung einer zentralen, dezentralen und lernenden Steuerung. Beim zentralen Ansatz berechnet eine zentrale Recheneinheit mithilfe eines Genetischen Algorithmus rollierende sowie ereignisgesteuerte Ablaufpläne für die Produktion. Beim dezentralen Ansatz verhandeln Produkte mit Ressourcen in einem auktionsbasierten Multiagentensystem um die Ressourcenallokation, wobei das Contract Net Protocol und Erstpreisauktionen Anwendung finden. Der lernende Ansatz nutzt Proximal Policy Optimization aus dem Bereich des Deep Reinforcement Learnings zum Sammeln interaktionsbasierter Erfahrungen, aus denen er die Steuerung per „trial-and-error“ lernt. Die Evaluation erfolgt anhand eines realitätsorientierten Produktionsbeispiels unter variierenden Produktionsszenarien. Aus der Evaluation lassen sich zwei Handlungsempfehlungen für die Steuerungsentwicklung in Modularen Produktionen ableiten: Zum einen empfiehlt diese Arbeit durchgängig die Entwicklung lernender Steuerungen, vor allem wenn höhere Komplexitätsgrade in der Produktionskonfiguration vorliegen. Zum anderen ist eine Verzahnung von Planung und Steuerung bei der Umsetzung neuer modularer Produktionssysteme von elementarer Bedeutung, da dadurch eine optimale Abstimmung der Modularen Produktion mit ihrer Steuerung erfolgen kann.

Abstract

In the context of the fourth industrial revolution, the trend towards individualization, and the legally driven development of new drive concepts, the production paradigm of Modular Production with decoupled stations and flexible material transport by an automated guided vehicle system has emerged in the automotive industry. To utilize the efficiency advantages of Modular Production there is a need for action in the development of effective control systems. This work addresses this need by developing a centralized, decentralized, and learning control system. In the centralized approach, a central computing unit calculates schedules for the production system in a rolling and event-driven manner using a Genetic Algorithm. In the decentralized approach, products negotiate with resources in an auction-based multi-agent system for resource allocation using the Contract Net Protocol and first-price auctions. The learning control system applies Proximal Policy Optimization from Deep Reinforcement Learning to collect interaction-based experiences, from which it learns scheduling via trial-and-error. The evaluation is conducted based on a reality-oriented production example under varying production conditions. Two recommendations for the development of control systems in Modular Production can be derived from the evaluation: First, this work recommends the development of learning control systems, especially when higher complexity levels are present in the production configuration. Second, an interlocking of planning and control is of elementary importance in the implementation of new modular production systems, because this allows the Modular Production and its control to be optimally coordinated.

Inhaltsverzeichnis

1 Einführung **1**
1.1 Modulare Produktion . . . 1
1.2 Ausgangspunkt und Zielsetzung . . . 3
1.3 Beiträge der Arbeit . . . 5
1.4 Aufbau der Arbeit . . . 6

2 Grundlagen zur Modularen Produktion und ihrer Steuerung **9**
2.1 Modulare Produktion . . . 9
2.1.1 Konzepte der Modularen Produktion . . . 10
2.1.2 Darstellung und Funktionsweise der Modularen Produktion . . . 12
2.1.3 Logistische Zielgrößen und Kennzahlen . . . 15
2.2 Steuerung der Modularen Produktion . . . 19
2.2.1 Produktionsplanung und -steuerung . . . 19
2.2.2 Entscheidungskategorien zur Steuerung der Modularen Produktion . 22
2.2.3 Scheduling als Entscheidungsprozess . . . 24
2.2.4 Steuerungsparadigmen und -architekturen . . . 29
2.3 Steuerung im Kontext der Industrie 4.0 . . . 32
2.3.1 Wandel von zentralen zu dezentralen Steuerungssystemen . . . 32
2.3.2 Multiagentensysteme . . . 35
2.3.3 Lernende Agenten mittels Reinforcement Learning . . . 38
2.3.4 Virtuelle Inbetriebnahme von Steuerungssystemen . . . 40
2.4 Zusammenfassung der Steuerung der Modularen Produktion . . . 41
2.4.1 Zentrale, dezentrale und lernende Steuerungssysteme . . . 42
2.4.2 Problembeschreibung, Anforderungen und Annahmen . . . 43

3 Zentraler Ansatz: Rollierendes Scheduling **47**
3.1 Stand der Forschung zentraler Steuerungsansätze . . . 47
3.1.1 Überblick zu zentralen Steuerungsansätzen . . . 47
3.1.2 Analyse ausgewählter zentraler Steuerungsansätze . . . 48
3.1.3 Schlussfolgerungen für die Entwicklung der zentralen Steuerung . . . 53
3.2 Methode: Genetische Algorithmen . . . 55
3.3 Konzeptionierung des rollierenden Schedulings . . . 60
3.3.1 Auftragseinlastung . . . 61
3.3.2 Stationsmanagement, Auftragsrouting und Flottenmanagement . . . 62
3.3.3 Störungsmanagement . . . 74
3.4 Zusammenfassung und Einordnung des Ansatzes . . . 77

4 Dezentraler Ansatz: Auktionsbasiertes Multiagentensystem **79**
4.1 Stand der Forschung dezentraler Steuerungsansätze . . . 79
4.1.1 Überblick zu dezentralen Steuerungsansätzen . . . 79

4.1.2 Analyse ausgewählter dezentraler Steuerungsansätze 81
4.1.3 Schlussfolgerungen für die Entwicklung der dezentralen Steuerung . . 87
4.2 Methode: Contract Net Protocol und Erstpreisauktion 89
4.3 Konzeptionierung des auktionsbasierten Multiagentensystems 92
4.3.1 Auftragseinlastung 94
4.3.2 Stationsmanagement, Auftragsrouting und Flottenmanagement . . . 95
4.3.3 Störungsmanagement 101
4.4 Zusammenfassung und Einordnung des Ansatzes 102

5 Lernender Ansatz: Deep Reinforcement Learning-Agent 105
5.1 Stand der Forschung lernender Steuerungsansätze 105
5.1.1 Überblick zu lernenden Steuerungsansätzen 105
5.1.2 Analyse ausgewählter lernender Steuerungsansätze 107
5.1.3 Schlussfolgerungen für die Entwicklung der lernenden Steuerung . . . 112
5.2 Methode: Deep Reinforcement Learning 114
5.2.1 Grundlagen zum Reinforcement Learning 114
5.2.2 Policy-Gradient-Methoden 119
5.2.3 Actor-Critic Proximal Policy Optimization 122
5.3 Konzeptionierung des Deep Reinforcement Learning-Agenten 129
5.3.1 Agentendesign 130
5.3.2 Lösen der fünf Entscheidungskategorien 140
5.4 Zusammenfassung und Einordnung des Ansatzes 143

6 Simulationsbasierte Evaluation der Steuerungsalgorithmen 145
6.1 Evaluationsbeispiel aus der Automobilindustrie 145
6.2 Implementierung der Algorithmuskonzepte 149
6.2.1 Modulare Produktion und Schnittstelle 149
6.2.2 Rollierendes Scheduling 154
6.2.3 Auktionsbasiertes Multiagentensystem 157
6.2.4 Deep Reinforcement Learning-Agent 159
6.2.5 Funktionalitätsprüfung mittels Plausibilitätstests 160
6.3 Ansatzspezifische Experimente 161
6.3.1 Rollierendes Scheduling 162
6.3.2 Auktionsbasiertes Multiagentensystem 168
6.3.3 Deep Reinforcement Learning-Agent 170
6.4 Vergleichende Experimente im Evaluationsbeispiel 173
6.4.1 Optimalität ohne Störungen 174
6.4.2 Robustheit gegenüber Störungen 178
6.5 Generalisierbarkeit auf erweiterte Evaluationsbeispiele 182
6.5.1 Dritter Produkttyp 182
6.5.2 Dritter Produkttyp und Zusatzstation 184
6.6 Zusammenfassung und Handlungsempfehlungen 187
6.6.1 Ansatzspezifische Erkenntnisse 187
6.6.2 Ergebniszusammenfassung 189
6.6.3 Handlungsempfehlungen für die Steuerungsentwicklung 190

7 Fazit und Ausblick **193**

Anhang

A Grundlagen **199**
A.1 Framework zur Virtuellen Inbetriebnahme 199

B Implementierung **201**
B.1 Verwaltetes Wissen im zentralen Ansatz 201
B.2 Verwaltetes Wissen im dezentralen Ansatz 202

C Ansatzspezifische Experimente **203**
C.1 Zentral: Auslegung des Ermittlungszeitraums mit Nullzeitannahme 203
C.2 Zentral: 30 im Vergleich zu fünf Simulationen 205
C.3 Dezentral: Auslegung der Vorausschau 206
C.4 Dezentral: 30 im Vergleich zu fünf Simulationen 207
C.5 Lernend: PPO Lernkurve mit Action Masking 208
C.6 Lernend: PPO Lernkurve für unterschiedliche Anzahl von Umgebungen . . . 209
C.7 Lernend: Durchsatzverläufe für unterschiedliche Anzahl von Umgebungen . . 210
C.8 Lernend: 30 im Vergleich zu fünf Simulationen 211

D Vergleichende Experimente **213**
D.1 Durchsatzverläufe im statischen Fall 213
D.2 Simulations-Varianz durch unterschiedliche Störungs-Seeds, Verfügbarkeit 0.98 214
D.3 Durchlaufzeiten im dynamischen Fall, Verfügbarkeit 0.98 215
D.4 Simulations-Varianz durch unterschiedliche Störungs-Seeds, Verfügbarkeit 0.90 216
D.5 Durchsatzverläufe im dynamischen Fall, Verfügbarkeit 0.90 217
D.6 Lernend: Generalisierung der gelernten Modelle 218

E Generalisierende Experimente **219**
E.1 Beispiel 2: Stationsauslastungen, Verfügbarkeit 0.98 219
E.2 Beispiel 2: Durchsatzverläufe und Durchlaufzeiten, Verfügbarkeit 0.98 221
E.3 Beispiel 3: Durchsatzverläufe und Durchlaufzeiten, Verfügbarkeit 0.98 223

Abkürzungsverzeichnis **225**

Symbolverzeichnis **229**

Literaturverzeichnis **239**

1 Einführung

Die Produktionsindustrie entwickelte sich im Laufe der letzten Jahrhunderte stetig weiter. Die größten Treiber waren hierbei die industriellen Revolutionen. Die erste industrielle Revolution betraf die Erfindung der Dampfmaschine Ende des 18. Jahrhunderts. Die zweite Revolution beinhaltete die Entwicklung arbeitsteiliger Massenproduktion mittels elektrischer Energie und wurde durch Henry Fords Fließbandfertigung in der Automobilproduktion 1913 geprägt. Mitte der siebziger Jahre fand ein fließender Übergang in die dritte Revolution durch den Aufschwung der Mikroelektronik statt [1, S. 5f.], [2]. Als vierte industrielle Revolution bezeichnet der Begriff Industrie 4.0 die intelligente Vernetzung von Produkten, Ressourcen und Abläufen mithilfe von Informations- und Kommunikationstechnologie [3, S. 20]. Im Projekt *Industrie 4.0* unterstützt das Bundesministerium für Bildung und Forschung (BMBF) die deutsche Industrie bei der vierten Revolution. Die Regierung verfolgt das Ziel, eine Individualisierung der Produkte unter den Bedingungen einer flexibilisierten Produktion zu fördern [4]. Dadurch unterstützt das BMBF den Wandel von der Fließfertigung hin zur flexibleren Organisationsform der Modularen Produktion (MP).

1.1 Modulare Produktion

Henry Fords Grundprinzip der Fließfertigung hat seit seiner Einführung 1913 einen hohen Stellenwert und findet auch heute in der Automobilindustrie noch in einer adaptierten Form flächendeckend Anwendung. Die weiterentwickelte Form der Fließfertigung wird als Variantenfließfertigung bezeichnet, welche aufgrund der steigenden Zahl an Produktvarianten entwickelt wurde [5]. In einer Variantenfließfertigung ermöglichen Universalmaschinen mit automatisierten Werkzeugwechseln sowie flexibel ausgebildete Werker die Montage unterschiedlicher Produktvarianten in einer Fertigungslinie [6]. Die einzelnen Arbeitsstationen sind entlang dieser Linie organisiert. Die Produkte durchlaufen die Stationen in einem kontinuierlichen oder aussetzenden Fluss. Bei kontinuierlichem Fluss bewegt sich das Fließband mit einer kontinuierlichen Geschwindigkeit. Bei aussetzendem Fluss stoppt das Fließband während der Bearbeitung [7, S. 8f.]. Abbildung 1.1a zeigt das Fließprinzip. Die Produkte bewegen sich in beliebiger Reihenfolge über das Band. Dabei existieren in einer Fahrzeugmontage pro Produkttyp, z.B. einem Audi A4, zahlreiche Konfigurationsvarianten, wie Abbildung 1.1b durch die Darstellung der jeweiligen Farbstufen eines Produkttyps verdeutlicht.

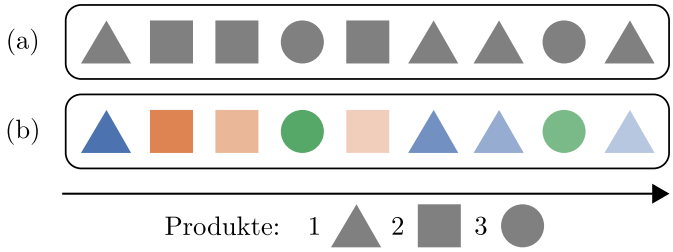


Abbildung 1.1: Prinzip der Variantenfließfertigung mit unterschiedlichen formcodierten Produkten 1, 2 und 3 (a) und jeweils verschiedenen farbcodierten Varianten der Produkte (b), Quelle: Eigene Darstellung in Anlehnung an Boysen [7, S. 11].

Die Bearbeitungsvorgänge der Produkttypen und -varianten unterscheiden sich sowohl hinsichtlich der bereitzustellenden Materialien als auch hinsichtlich der jeweiligen Bearbeitungszeit. Dies stellt die Produktion vor zwei Herausforderungen: die rechtzeitige Materialbereitstellung an der Linie durch die Logistik und die schwankenden Bearbeitungszeiten an den Stationen, was als Fertigungszeitspreizung bezeichnet wird [8, S. 6]. Die sogenannte Reihenfolgeplanung versucht die Fertigungszeitspreizung auszugleichen, indem die Reihenfolge der Produktvarianten auf der Linie gesteuert wird. Sie zielt auf eine ausgeglichene Auslastung der Stationen ab, kann aber aufgrund der Zeitspreizung kein Optimum darstellen [9].

Auch die Variantenfließfertigung stößt wegen der weiter steigenden Produkt- und Variantenvielfalt an ihre Grenzen. Wesentliche Gründe für diese Vielfalt sind der Trend der Individualisierung [10, S. 15] sowie die Entwicklung neuer Antriebskonzepte, welche von gesetzlich festgelegten CO_2-Grenzwerten für Fahrzeugflotten getrieben wird [11], [12]. Die zunehmende Spreizung der Taktzeiten unterschiedlicher Produkttypen und -varianten führt zu Intervallen an Arbeitsstationen, in denen die Stationen entweder überlastet oder nicht vollständig ausgelastet sind. Es resultieren Ineffizienzen, weil Produktionszeit in Form vorhandener, nicht genutzter Kapazität verschwendet wird [13]. Hinzu kommen Herausforderungen wie Stationsstörungen, die Fließfertigungssysteme zum Stillstand bringen, sowie Marktdynamiken und dadurch schwankende und sich verändernde Kundenbedarfe, denen ein produzierendes Unternehmen unterliegt [14], [10, S. 14]. An diese Gegebenheiten sind die Produktionssysteme für eine effiziente Produktion anzupassen. Sie erfordern Wandlungsfähigkeit, Robustheit und Flexibilität [15]. Unter diesen Gesichtspunkten, unterstützt durch das Projekt *Industrie 4.0*, entwickelte sich das Paradigma der MP [16]–[18].

Kern [18, S. 134f.] definiert die MP als Organisationsform für Produktionssysteme mit entkoppelt angeordneten Stationen. Der Transport der zu fertigenden Produkte erfolgt auf einem flexiblen Fördersystem. Das flexible Fördersystem fährt die Stationen entsprechend der Fertigungsabläufe der geladenen Produkte bedarfsabhängig und zustandsbasiert an. Dabei definieren die Vorgabezeiten der Bearbeitungsvorgänge die Verweilzeiten von Fördereinheit und Produkt an einer Station.

Die Entkopplung der Stationen löst das Problem der Ineffizienzen durch die Fertigungs-

zeitspreizung, indem ein Produkt an einer Station so lange wie nötig verweilen kann, ohne dabei den gesamten Produktionsfluss zu blockieren. Dadurch agiert die MP bedarfsgerecht und abgestimmt auf die zu fertigenden Produkte [13]. Das für die MP notwendige flexible Fördersystem wird häufig durch ein Fahrerloses Transportsystem (FTS) abgebildet, welches aus einer Flotte autonom agierender Fahrerloser Transportfahrzeuge (FTF) besteht, die die Produkte aufnehmen und transportieren [19]. Die erzielbare Effizienzverbesserung ist in der MP allerdings nur zu erreichen, wenn die Produktionssteuerung die Produkte, Stationen und FTFs mittels effektiver Entscheidungsstrategien bedarfsgerecht und optimal koordiniert, wobei sich optimal beispielsweise auf eine Durchsatzmaximierung bezieht. Kern [18, S. 130] schlussfolgert hierbei Handlungsbedarf in der Konzeption derartiger Steuerungssysteme.

1.2 Ausgangspunkt und Zielsetzung

Die Entwicklung einer MP-Steuerung kann sich hierbei an Steuerungssystemen für verwandte Produktionsarten orientieren, da modulare Produktionskonzepte nicht ausschließlich in der Automobilindustrie einen Aufschwung erleben. Verwandt ist das MP-Konzept beispielsweise mit Organisationsformen wie der Werkstattfertigung (engl.: job-shop production) und sogenannten flexiblen Fertigungssystemen (engl.: flexible manufacturing systems (FMS)). Eine Werkstattfertigung zeichnet sich hierbei ähnlich zur MP durch eine verteilte Maschinenanordnung aus. Nach dem Verrichtungsprinzip werden Maschinen gleicher Fähigkeiten zu beispielsweise einer Dreherei oder Fräserei gruppiert. Gefertigt wird in Losen, die gemeinsam an einer Maschine abgearbeitet und dann meist durch Fertigungsmitarbeiter weiter transportiert werden [20, S. 10-11]. Werkstattfertigungssysteme finden beispielsweise in komplexen Halbleiterfertigungen Anwendung, wo im Kontext der Industrie 4.0 erhebliche Optimierungspotenziale im Bereich des Schedulings gesehen werden [21]. Flexible Fertigungssysteme verbinden die Flexibilität der Werkstattfertigung mit der Effizienz der Fließfertigung. Sie werden als vollständig automatisierter computergesteuerter Verbund aus einem Materialflusssystem und Computerized Numerical Control (CNC)-Maschinen definiert [22]. Die CNC-Maschinen sind dabei in der Lage, im Rahmen gegebener Produktfamilien ähnliche geometrische Operationen an Produkten auszuführen, wodurch auf Produktänderungen und wandelnde Märkte reagiert werden kann [23]. Sowohl Werkstattfertigungssysteme als auch flexible Fertigungssysteme erfordern wie die MP eine entsprechende optimale Steuerungslogik, damit die anvisierten Flexibilitätsvorteile der jeweiligen Produktionsform nutzbar sind. Deshalb bieten Steuerungslösungen für beide Produktionsarten Anhaltspunkte für die Steuerung der MP.

Welche Art der Steuerung sich für die MP am besten eignet, hängt dabei einerseits davon ab, welche logistischen Zielgrößen zu optimieren sind. Andererseits spielt auch die Produktionskonfiguration und -größe eine entscheidende Rolle. Dies zeigt sich auch in der Literatur für mit der MP verwandte Steuerungsprobleme, wobei je nach Produktionsbedingungen und

-zielen unterschiedliche Steuerungsparadigmen bessere Ergebnisse erzielen [24]–[27].

Neben der Existenz verwandter Produktionsarten zeichnet sich die MP auch innerhalb der Automobilindustrie durch ihre Diversität aus. Konzepte unterscheiden sich anhand der zu fertigenden Produkte bzw. Bauteile. Eine erste MP-Umsetzung der Audi AG ist in Abbildung 1.2 zu sehen. Die Abbildung zeigt ein FTF, welches für den Materialtransport zwischen modularen Stationen in einer E-Achsproduktion im Einsatz ist. Dabei sind die Produkte durchsatzoptimal mithilfe der zur Verfügung stehenden Stationen und Fahrzeuge zu produzieren, was aktuell von einfach gehaltenen Steuerungsregeln adressiert wird. An diese reale MP-Umsetzung ist die vorliegende Arbeit in Kooperation mit der Audi AG angelehnt.

Abbildung 1.2: FTF im Einsatz in einer MP der Audi Hungaria Zrt., Quelle: Mediencenter ©Audi AG [28].

Um den Handlungsbedarf im Bereich effektiver Steuerungsalgorithmen für die MP in der Automobilindustrie anzugehen, stellen sich drei aufeinander aufbauende Forschungsfragen:

1. **Forschungsfrage**: Welche Steuerungsarchitekturen für die MP stehen im Stand der Forschung zur Verfügung?

2. **Forschungsfrage**: Wie lassen sich geeignete Steuerungsarchitekturen in der MP umsetzen, um eine gegen Ressourcenstörungen robuste, in Bezug auf den Durchsatz möglichst optimale sowie realitätsnahe Steuerung zu konzipieren?

3. **Forschungsfrage**: Welche Steuerungsarchitektur ist für die MP in Abhängigkeit der Produktionskonfiguration sowie der Beschaffenheit von Ressourcenstörungen am besten geeignet?

Die Beantwortung der drei Forschungsfragen verfolgt das Ziel, realistische Handlungsempfehlungen für die Umsetzung von Steuerungsalgorithmen in der MP in Abhängigkeit der Produktionskonfiguration, der damit verbundenen Problemgröße und -komplexität sowie der in der Produktion auftretenden Störungsdynamik abzuleiten. Die Handlungsempfehlungen sollen Anhaltspunkte geben, unter welchen Bedingungen welches Steuerungsparadigma in

der MP zu bevorzugen ist. Die Güte der Steuerungsentscheidungen der einzelnen Ansätze wird hierfür primär vom mittleren Durchsatz D_m bestimmt, welchen die Steuerung im Vergleich zum erreichbaren Maximum D_{max} erzielt.

1.3 Beiträge der Arbeit

Die vorliegende Arbeit beantwortet die drei Forschungsfragen mit dem Ziel realistischer Handlungsempfehlungen für die Umsetzung von MP-Steuerungen. Hierzu erfasst sie das konkrete Steuerungsproblem der MP, entwickelt eine zentrale, eine dezentrale sowie eine lernende Steuerung und analysiert die Verfahren unter variierenden Rahmenbedingungen. Dabei leistet sie die folgenden Beiträge zum Stand der Forschung:

- Erfassung der MP-Entscheidungskategorien: Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement – Abschnitt 2.2.2,
- Ableitung geeigneter Steuerungsparadigmen für die fünf Entscheidungskategorien: zentrales, dezentrales und lernendes Paradigma – Abschnitte 2.2.4, 2.3.2 und 2.3.3,
- Entwicklung eines zentralen Algorithmus zur Steuerung der MP mithilfe eines Genetischen Algorithmus: Rollierendes Scheduling; Schedule-Berechnung zeitlich periodisch, bei Auftragseinlastung, Störungsstart und Störungsende – Abschnitt 3.3, [29]–[31],
- Entwicklung eines dezentralen Algorithmus zur Steuerung der MP auf Basis des Contract Net Protocols und Erstpreisauktionen: Auktionsbasiertes Multiagentensystem; Auktionsstart aus Stationssicht bei freier Kapazität in absehbarer Zukunft; Produktionsaufträge bieten für Kapazität – Abschnitt 4.3, [32], [33],
- Entwicklung eines lernenden Algorithmus zur Steuerung der MP mithilfe von Proximal Policy Optimization: Deep Reinforcement Learning-Agent; ereignisbasierte Entscheidungen durch effizientes Action-, State- und Reward-Design – Abschnitt 5.3, [34]–[36],
- Realitätsnahe Implementierung der drei Steuerungsalgorithmen mithilfe des Konzepts der Virtuellen Inbetriebnahme für eine vergleichende Evaluation unter jeweils identischen Rahmenbedingungen – Abschnitt 6.2, [37],
- Evaluation und Vergleich der Steuerungsalgorithmen in Abhängigkeit der Beschaffenheit von Ressourcenstörungen in der MP – Abschnitt 6.4.2,
- Evaluation und Vergleich der Steuerungsalgorithmen in Abhängigkeit der Produktionskonfiguration der MP – Abschnitt 6.5,
- Handlungsempfehlungen für die Umsetzung von Steuerungsalgorithmen für die MP – Abschnitt 6.6.3.

Im Kern befasst sich die vorliegende Arbeit mit der Steuerung der MP in der Automobilindustrie. Die Entwicklung der drei Steuerungsalgorithmen orientiert sich hierzu an verwandten Produktionsarten. Dabei zeichnen sich die drei Steuerungssysteme durch ihre praxisnahe Implementierung sowie die Adressierung aller fünf Entscheidungskategorien der MP aus. Die generierten Erkenntnisse der vorliegenden Arbeit gehen den Handlungsbedarf im Bereich effektiver MP-Steuerungen in der Automobilindustrie an und stellen eine umfassende Grundlage für die zukünftige Steuerungsentwicklung dar. Schließlich sind die Algorithmen auf verwandte Produktionsarten transferierbar, wobei Adaptionen auf die jeweilige Problemstellung notwendig sind.

1.4 Aufbau der Arbeit

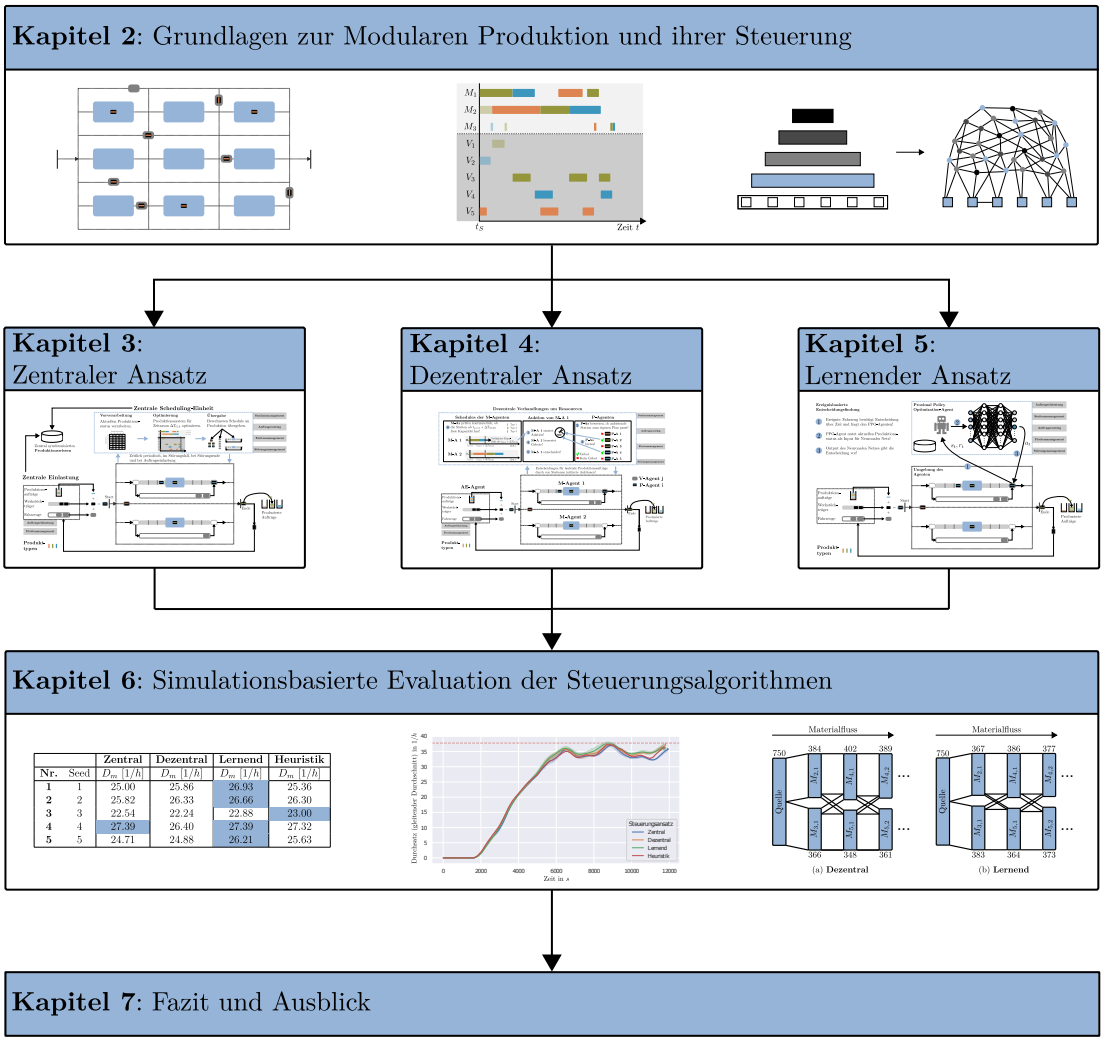
Kapitel 2 führt die Grundlagen zum Verständnis der vorliegenden Arbeit ein. Dabei leitet das Kapitel zum Konzept und der Funktionsweise der MP hin und beschreibt wesentliche logistische Kennzahlen. Anschließend setzt das Kapitel den Fokus auf die Steuerung der MP, führt die zu lösenden Entscheidungskategorien ein und gibt eine Übersicht zu verfügbaren Steuerungsparadigmen. Daraufhin betrachtet die Arbeit die Steuerung im Kontext von Industrie 4.0 und beschreibt eine Entwicklung hin zu verhandelnden und lernenden Agentensystemen für die Entscheidungsfindung in Produktionssystemen. Nach Vorstellung der Virtuellen Inbetriebnahme schließt das Kapitel mit einer Zusammenfassung der Problemstellung ab und definiert Anforderungen an ein Steuerungssystem der MP.

Abgeleitet aus Kapitel 2 bilden Kapitel 3, Kapitel 4 und Kapitel 5 den Kern der Arbeit. Sie entwickeln jeweils einen eigenständigen Steuerungsalgorithmus. Dabei sind die Kapitel identisch aufgebaut: Zuerst legen sie den jeweiligen Stand der Forschung dar und leiten ein Grobkonzept zur Steuerung ab. Danach erfolgt die Beschreibung der gewählten methodischen Grundlagen. Es folgt die Vorstellung des Feinkonzepts des jeweiligen Algorithmus. Eine Zusammenfassung und Einordnung in die Anforderungen schließt das jeweilige Algorithmuskapitel ab. Kapitel 3 entwickelt hierbei einen zentralen Steuerungsansatz auf Basis eines Genetischen Algorithmus und verwendet das Prinzip des rollierenden Schedulings. Kapitel 4 leitet ein dezentrales Multiagentensystem ab, wobei das Contract Net Protocol sowie Erstpreisauktionen zur auktionsbasierten Entscheidungsfindung im Einsatz sind. Kapitel 5 entwickelt, den Kern der Arbeit abschließend, einen lernenden Deep Reinforcement Learning-Agenten, der die Steuerung mittels Proximal Policy Optimization erlernt und das Gelernte anschließend umsetzt.

Kapitel 6 beschreibt schließlich ein Evaluationsbeispiel aus der Automobilindustrie sowie die realitätsnahen Implementierungen der drei Steuerungsalgorithmen. Nach ansatzspezifischen Experimenten vergleicht das Kapitel die Algorithmen unter jeweils identischen Rahmenbedingungen hinsichtlich ihrer Optimalität ohne Störungen, der Robustheit gegenüber

Störungen sowie der Generalisierbarkeit auf erweiterte Evaluationsbeispiele. Auf den Ergebnissen basierende Handlungsempfehlungen schließen das Kapitel ab.

In Kapitel 7 zieht die vorliegende Arbeit ein Fazit, beantwortet die Forschungsfragen und gibt einen Ausblick auf zukünftige Forschungsrichtungen.



		Zentral	Dezentral	Lernend	Heuristik
Nr.	Seed	D_m [1/h]	D_m [1/h]	D_m [1/h]	D_m [1/h]
1	1	25.00	25.86	26.93	25.36
2	2	25.82	26.33	26.66	26.30
3	3	22.54	22.24	22.88	23.00
4	4	27.39	26.40	27.39	27.32
5	5	24.71	24.88	26.21	25.63

Abbildung 1.3: Aufbau der Arbeit.

2 Grundlagen zur Modularen Produktion und ihrer Steuerung

Dieses Kapitel führt die zum Verständnis der Arbeit notwendigen Grundlagen ein. Abschnitt 2.1 leitet zum Konzept der MP hin und beschreibt die wesentlichen logistischen Kennzahlen. Abschnitt 2.2 setzt den Fokus anschließend auf die Steuerung der MP, führt die zu treffenden Entscheidungen ein und beschreibt Steuerungsparadigmen, die zur Lösung des Steuerungsproblems angewendet werden können. Danach betrachtet Abschnitt 2.3 die Steuerung im Kontext von Industrie 4.0 und beschreibt die Entwicklung von zentral organisierten Steuerungsansätzen hin zu verhandelnden und lernenden Agentensystemen für die Entscheidungsfindung. Nachdem die Virtuelle Inbetriebnahme als Methode zur Entwicklung und Evaluation der Steuerung dargestellt wurde, geht Abschnitt 2.4 zusammenfassend auf eine geeignete Problemstellung zur Beantwortung der Forschungsfragen ein. Indem das gesamte Kapitel potenzielle Steuerungsparadigmen für die MP ableitet, leistet es bereits einen Beitrag zur Beantwortung der ersten Forschungsfrage aus Abschnitt 1.2.

2.1 Modulare Produktion

Für produzierende Unternehmen ist die Produktion der Schlüssel, ihre Produkte in hoher Qualität zu wettbewerbsfähigen Preisen anbieten zu können, indem die vorhandenen Produktionsressourcen effektiv genutzt werden. Die Produktion erfüllt deshalb die Aufgabe, mittels Arbeitskraft, Material, Energie, Kapital und Information entsprechende Produkte zu produzieren. Hierzu schließt sie die eng vernetzten Disziplinen Konstruktion, Arbeitsvorbereitung, Fertigung und Montage mit ein [38, S. 15-25]. Die Konstruktion betrifft das Planen, Konzipieren, Entwerfen und Ausarbeiten zu produzierender Produkte [39, S. 128]. Da sich diese Arbeit auf die Produktion bereits konstruierter und entwickelter Produkte konzentriert, geht sie an dieser Stelle nicht näher auf die Konstruktion ein. Zur Arbeitsvorbereitung zählt die Produktionsplanung und -steuerung. Sie wird als wesentliche Steuerungsgrundlage in Abschnitt 2.2.1 diskutiert. Fertigung und Montage sind hingegen Themen dieses Abschnitts. Zusammen bilden sie den Kernbereich der Wertschöpfung produzierender Unternehmen [40, S. 101]. Fertigung bezeichnet nach DIN 8550 die Herstellung von Einzelteilen aus Rohmaterial [41]. Montage wiederum beschreibt die Vorgänge zum Zusammenbau geometrisch

bestimmter Einzelteile und Baugruppen zu funktionsfähigen Produkten [42, S. 163][1]. In der Fertigung ist der Automatisierungsgrad i.d.R. höher als in der Montage, die sich hingegen durch eine höhere Komplexität auszeichnet. Als Ausnahme gilt beispielsweise die Aggregatemontage im Fahrzeugbau, die häufig zumindest teilautomatisiert arbeitet [40, S. 124]. Die in dieser Arbeit betrachteten modularen Produktionskonzepte finden sowohl für Fertigung als auch für Montage Anwendung, wie Abschnitt 2.1.1 zeigen wird. In dieser Arbeit meint Produktion als Überbegriff deshalb in Abhängigkeit vom Kontext entweder Fertigung oder Montage. Zur Bezeichnung der modularen Organisationsform der Produktion wird folglich der allgemeingültigere Begriff der MP gewählt. Abschnitt 2.1.1 fasst aktuelle modulare Produktionskonzepte aus Forschung und Technik der Automobilindustrie zusammen und legt damit die Grundlage für die in dieser Arbeit betrachtete Art der MP, die in Abschnitt 2.1.2 vorgestellt wird. Abschnitt 2.1.3 schließt das Teilkapitel mit der Erläuterung wesentlicher Produktionskennzahlen in Bezug auf die MP ab.

2.1.1 Konzepte der Modularen Produktion

MP-Konzepte finden sowohl im Bereich der Forschung als auch im industriellen Umfeld der Automobilindustrie Anwendung. In Anlehnung an [18, S. 101-112] verschaffen die folgenden Absätze hierzu einen groben Überblick und suchen Gemeinsamkeiten in den untersuchten Konzepten. Es erfolgt keine detaillierte Konzeptanalyse, da als Kern der Arbeit die Steuerung der MP im Vordergrund steht.

Zahlreiche Forschungs- und Promotionsprojekte befassen sich mit der MP als Alternative zu Fließfertigungssystemen. Einen aktuellen Überblick zu den umfangreichsten Untersuchungen mit Fokus auf die Automobilindustrie gibt Tabelle 2.1.

Tabelle 2.1: Forschung im Bereich modularer Produktionssysteme.

Projektart	**Name**	**Beschreibung**
Forschungsprojekt	*ARENA2036*	Flexible Forschungsfabrik
Forschungsprojekt	*SmartFace*	Smart Micro Factory für Elektrofahrzeuge
Forschungsprojekt	*SmartBodySynergy*	Smarte Rohbauzellen
Forschungsprojekt	*freeMoVe*, *AIMFREE*	Wandlungsfähige Montage
Promotionsprojekt	Greschke	Matrixproduktion
Promotionsprojekt	Bochmann	Fertigung für variantenreiche Produkte
Promotionsprojekt	Kern	Modulare Automobilproduktion

Die Forschungsplattform *ARENA2036* (Active Research Environment for the Next Generation of Automobiles) befasst sich in Zusammenarbeit mit Industriepartnern wie Siemens, Bosch und BMW mit innovativen modularen Produktionskonzepten [43], [44, S. 353-356]. Die Modularität zeigt sich in frei verketteten Bearbeitungsstationen [45, S. 114-118], [46,

[1] In Anlehnung an die VDI 2860, welche sich aktuell in Überarbeitung befindet.

S. 381-391], [14], flexibel verbunden durch autonome Fahrzeuge eines FTS [47], [48, S. 84-91]. Ähnliche Produktionsansätze verfolgt das Forschungsprojekt *SmartFace* zur Fertigung von Elektrofahrzeugen. Ziel sind ebenfalls verteilte Fertigungszellen, die durch zellulare Transportsysteme versorgt und verbunden werden [49]–[51]. Außerdem befasst sich das Projekt mit Themen wie Programmplanung und Kommunikationsfähigkeit der Produktionsentitäten [52]. Im Projekt *SmartBodySynergy* wird die Integration elektrifizierter Fahrzeuge in einen bestehenden Karosseriebau untersucht. Durch Modularisierung der Rohbauzellen, einen automatisierten Materialtransport sowie ein flexibles Steuerungs- und Logistikkonzept soll diese Integration ermöglicht werden [53]–[55]. Die Projekte *freeMoVe* und *AIMFREE* entwickeln ebenfalls eine MP mit unverketteten Bearbeitungsstationen. Der Fokus liegt auf der Wandlungsfähigkeit der MP, wofür Bausteine wie mobile Roboter, flexible Zuführ- und Greiftechnik sowie ein Steuerungs- und Leitsystem entwickelt werden [56]–[60], [61, S. 26-33]. Die Stationsplanung orientiert sich an der für die Anwendung nötigen Prozesstechnik [62].

In seiner Dissertation entwickelt Greschke [16] das Konzept der sogenannten „Matrixproduktion“ und beschreibt sie als taktunabhängige modular aufgebaute Produktion, in der Bearbeitungsschritte stets von mehreren Stationen ausgeführt werden können. Dies soll die Robustheit und Flexibilität der Produktion fördern [16, S. 93-173]. Bochmann [17] konzentriert sich in seiner Doktorarbeit auf Aspekte wie Layoutplanung der modularen Stationen, Maschinenbelegungsplanung zur Produktionskoordination und Transportwegplanung, damit die Fahrzeuge die kürzestmöglichen Wege suchen [17, S. 98-122]. Kern [18] entwickelt ein Gesamtkonzept für Organisation, Planung und Steuerung einer manuellen MP am Beispiel der Fahrzeugmontage. Auch die Logistik wird in das Konzept mit eingebunden, was in den meisten Projekten keine Berücksichtigung findet [13], [63], [64].

Neben der Forschung befasst sich auch die Industrie mit der MP. Kuka Systems entwickelte mit ihrer „Matrixproduktion“ einen flexiblen Karosseriebau, wobei die Materialien für die einzelnen Bearbeitungszellen per FTF angeliefert werden [65]. Das Unternehmen Synaos, als Ausgründung des VW-Konzerns, legt seinen Entwicklungsfokus auf das Flottenmanagement derartiger FTS-Systeme für beispielsweise modulare Produktionssysteme [66]. Arculus hingegen erarbeitete in Kooperation mit der Audi AG eine modulare Versuchsanlage und befindet sich aktuell in der Umsetzung des ersten modularen Montagesystems am Audi-Standort Ingolstadt, welches den Materialtransport mit hauseigenen FTFs durchführt. Serva Transport Systems (Serva TS) fertigt ebenfalls eigens konzipierte FTFs. Die Fahrzeuge von Serva TS wurden in einem Produktionsprojekt zur Modularen Montage von E-Achsantrieben mit der Audi Hungaria Zrt. zum Einsatz gebracht.

Allen Projekten gemein ist das von Kern in [18, S. 134f.] definierte Grundkonzept der MP aus unverketteten Bearbeitungsstationen, dem Materialtransport durch ein FTS und die dadurch steigenden Anforderungen an die Steuerungslogik. Der Fokus der Projekte liegt hierbei eher auf Seite der Konzeptionierung und Planung des modularen Produktionssystems, weniger auf der jeweiligen Steuerung. Jedoch untersuchen manche der Arbeiten, wie

z.B. Bochmann [17] mittels zentraler Maschinenbelegungs- und Transportwegplanung oder Echsler Minguillon und Lanza [55] durch eine Kombination aus zentraler und dezentraler Steuerung, auch Lösungen für die in der MP zu treffenden Steuerungsentscheidungen. Dies betont die Existenz unterschiedlicher Steuerungsparadigmen, welche später in Abschnitt 2.2 vorgestellt werden. Weiterhin wird im Verlauf der Arbeit deutlich, woran es den bereits vorhandenen Steuerungsansätzen für die Lösung des Steuerungsproblems der vorliegenden Arbeit mangelt. Einen Vergleich unterschiedlicher Steuerungsparadigmen unter variierenden Rahmenbedingungen führen die vorhandenen Arbeiten ebenfalls nicht durch. Bevor jedoch dieser Handlungsbedarf für MP-Steuerungssysteme näher betrachtet wird, stellt der nächste Abschnitt zunächst die Funktionsweise der MP dar.

2.1.2 Darstellung und Funktionsweise der Modularen Produktion

Ansätze zur MP konzentrieren sich auf entkoppelte Bearbeitungsstationen, wobei ein FTS die zu fertigenden Produkte von Station zu Station organisiert. In den meisten Konzepten, wie beispielsweise bei Greschke [16] und Bochmann [17], befinden sich die entkoppelten Stationen in einer raster- bzw. matrixförmigen Anordnung. Das Transportnetzwerk für das FTS orientiert sich dabei an der rasterförmigen Stationsanordnung [18, S. 261]. In Kombination mit dem Aufbau der bei der Audi AG entwickelten modularen E-Achsproduktion, an die sich die praxisnahe Untersuchung dieser Arbeit anlehnt, lässt sich der in Abbildung 2.1 dargestellte schematische Aufbau einer MP ableiten. Hierbei kann die Bearbeitung an einer Station entweder automatisiert oder manuell erfolgen. Das dargestellte Konzept eignet sich deshalb sowohl für Fertigung als auch für Montage.

Bei der Quelle erfolgt die Auftragseinlastung auf einem Werkstückträger (WT). Die Fahrzeuge transportieren die Produkte auf ihren WTs von Station zu Station, damit die Bearbeitung der Produkte stattfinden kann. Sie bewegen sich hierzu auf einem definierten Wegenetz. Ist ein Produkt fertig bearbeitet, findet ein letzter Transport zur Senke statt, welche beispielsweise einem Verpackungsband für den Versand des Produktes entsprechen könnte. In Konzepten wie dem von Kern [18] bleibt ein FTF für den gesamten Produktionsablauf fest mit dem Produkt verbunden. Die manuelle Bearbeitung an der Fahrzeugkarosserie erfolgt direkt auf dem FTF, welches die Karosserie transportiert. Im Gegensatz dazu gibt das FTF in anderen Konzepten das Produkt auf seinem WT an der Station ab und kann in der Zwischenzeit andere Transportaufträge durchführen.

Unabhängig davon, ob ein FTF das Produkt an einer Station abgibt oder durch das gesamte Produktionssystem trägt, zeichnet sich die MP durch den flexiblen Takt aus, der auf der Entkopplung der Stationen beruht. Die Entkopplung ermöglicht ebenfalls mehr Flexibilität, weil keine expliziten Routen für das Produkt vorgegeben werden [13]. Um diese Flexibilität für beispielsweise Maschinenausfälle nutzen zu können, sind die entkoppelten Stationen und die zu fertigenden Produkte entsprechend aufeinander abzustimmen und auszuplanen.

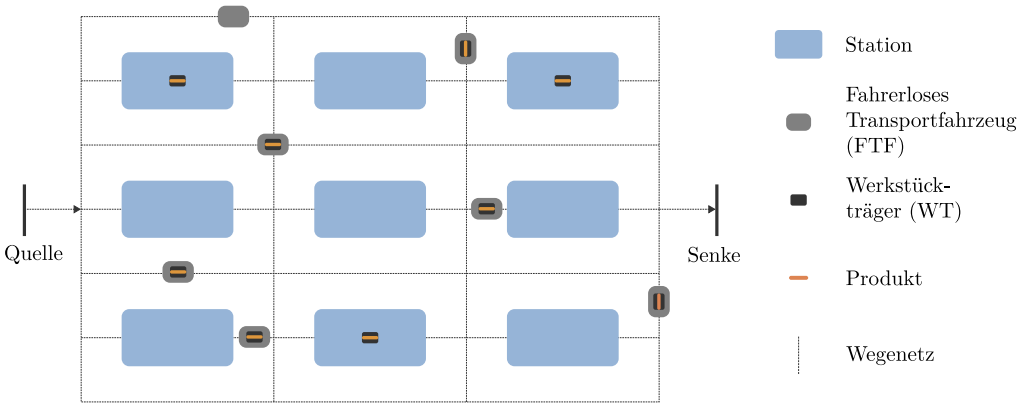


Abbildung 2.1: MP in einer rasterförmigen Anordnung von Bearbeitungsstationen und einem Wegenetz für den Materialtransport mittels FTFs, die die Produkte auf WTs von Station zu Station transportieren, Quelle: Eigene Darstellung in Anlehnung an [34].

Bochmann [17] weist den modularen Stationen hierzu Fähigkeiten zu, die sie Produkten zur Durchführung von Produktionsschritten anbieten können. Die Produktionsschritte der Produkte wiederum benötigen bestimmte Fähigkeiten, um bearbeitet werden zu können. Außerdem sind im Rahmen der technischen Machbarkeit des Vorranggraphen eines Produktes, also den Abhängigkeitsbeziehungen zwischen den Bearbeitungsschritten, auch Reihenfolgeänderungen der Schritte erlaubt. Auf Basis dieser variablen Reihenfolge und der Zuordnung zu Stationen anhand der Fähigkeiten ergeben sich für jedes Produkt mehrere mögliche Routen in einer gegebenen MP. In Bochmanns Konzept agieren die FTFs sowie eine funktionierende Echtzeitkommunikation zwischen den Produktionsentitäten als Enabler für diese flexible Routenführung [17, S. 94ff.].

Kern et al. [13] bezeichnen diese auf Fähigkeiten basierende Produktion als prozessbasierte MP, bezogen auf die Prozessarten, die eine Station durchführen kann. Diese Notation wird auch von Foith-Förster und Bauernhansl [46] sowie Foith-Förster et al. [14] verwendet, wenn sie von sogenannten Prozessmodulen sprechen. Zur Gewährleistung der Interoperabilität der Produktionsentitäten anhand gegebener Fähigkeiten werden Sprachen und Modelle entwickelt, in denen die Fähigkeiten einheitlich beschrieben werden, beispielsweise mittels Open Platform Communications Unified Architecture (OPC UA) [67]–[69].

Abbildung 2.2 stellt die oben beschriebenen Zusammenhänge zwischen Produkt und MP anhand eines Beispiels dar. Abbildung 2.2a zeigt ein modulares Produktionssystem, das aus vier Stationen $M := \{M_1, M_2, M_3, M_4\}$ besteht. Jede der Stationen bietet Fähigkeiten an, die einer Teilmenge der im Gesamtsystem vorhandenen Fähigkeiten $F := \{F_1, F_2, F_3, F_4\}$ entspricht. Die Fähigkeiten von Maschine M_1 sind beispielsweise $F_{M_1} := \{F_1, F_2\} \subseteq F$ und damit bietet sie „Drehen“ und „Kleben“ an. Das im Beispiel zu fertigende Produkt besteht aus vier Arbeitsschritten, dargestellt als Produktgraph[2] in Abbildung 2.2b. Die Reihenfolge

[2]Die Produktionsinformationen eines Produktes werden in dieser Arbeit als Produktgraph bezeichnet, weil sich die Bearbeitungszusammenhänge eines Produktes als Graph gut darstellen lassen und sich in der

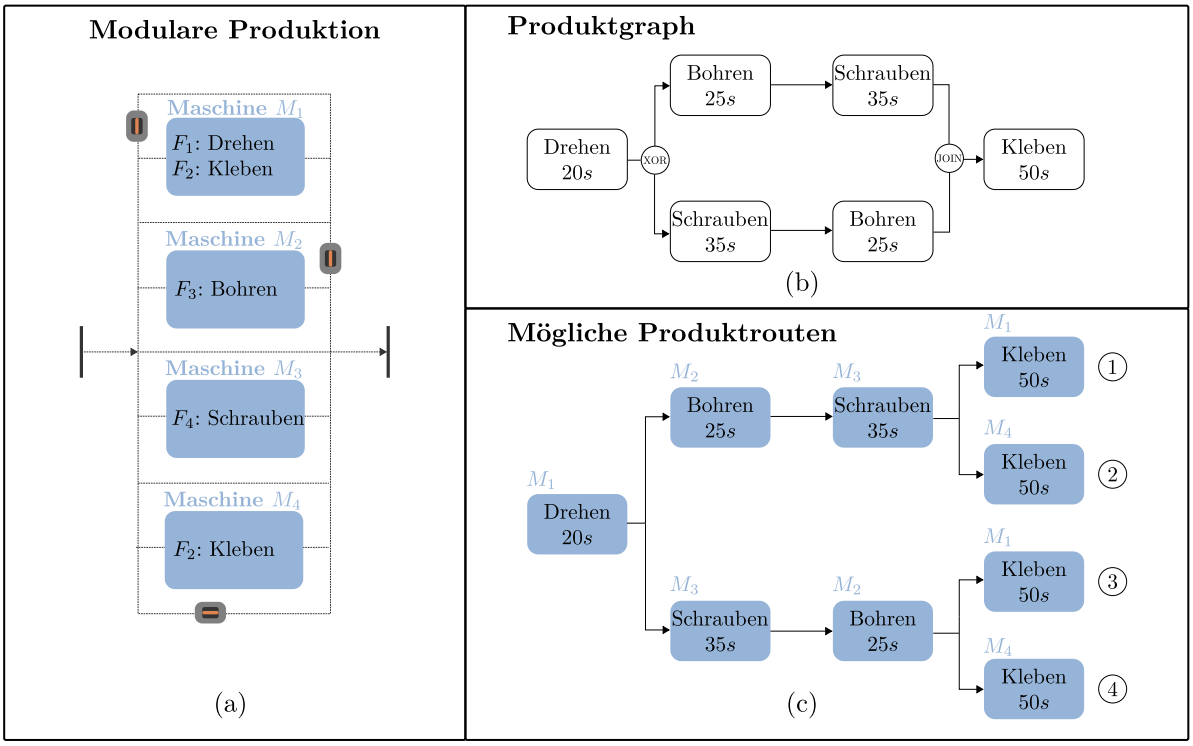


Abbildung 2.2: Modulares Produktionssystem mit vier Stationen, die jeweils bestimmte Fähigkeiten anbieten (a), in dem ein Produkt mit einem Produktgraphen bestehend aus vier Fertigungsvorgängen gefertigt werden kann (b), woraus sich ein gerichteter Baum für mögliche Routen im Produktionssystem ergibt (c), Quelle: Eigene Darstellung in Anlehnung an [29], [32].

der Bearbeitungsschritte zwei und drei und damit „Bohren“ und „Schrauben“ ist im Produktgraph flexibel, was als Reihenfolgeflexibilität bezeichnet wird [70, S. 11], [71]. Außerdem ist zu erkennen, dass die Bearbeitungszeiten nicht einem festen Takt entsprechen, sondern zwischen 20s und 50s variieren, analog zur bereits in Abschnitt 1.1 eingeführten Fertigungszeitspreizung. Beim Abgleich der für die Produktionsvorgänge vom Produkt benötigten Fähigkeiten mit dem Fähigkeitenangebot der vier Stationen ergibt sich die Zuordnung von Vorgang zu Station. Kommt für eine Operation mehr als eine Station infrage, beispielsweise für die Operation „Kleben“ die Stationen M_1 und M_4, wird dies als Operationsflexibilität bezeichnet [70, S. 11][71]. Aus Reihenfolge- und Operationsflexibilität lassen sich für das gegebene Produkt vier mögliche Routen durch die MP ableiten, die in Abbildung 2.2c in einer Baumstruktur dargestellt sind. Die Produktionssteuerung hat dabei die Aufgabe, sich für eine dieser Routen zu entscheiden. Jeder Verzweigungspunkt in der gerichteten Baumstruktur stellt hierfür eine Entscheidung dar. Die Summe der getroffenen Entscheidungen an Verzweigungspunkten ergibt die produktspezifische Route.

Unter der Annahme, dass sich beispielsweise 30 Produkte gleichzeitig im System befinden,

Literatur durchgesetzt hat. Der zu produzierende Produktionsauftrag repräsentiert die im Produktgraph dargestellten Informationen. Die Arbeit verwendet Produkt, Produktgraph und Produktionsauftrag als Synonyme.

für die jeweils 10 Produktionsschritte mit je einer Auswahl von zwei Stationen notwendig sind, ergeben sich $2^{10} = 1024$ unterschiedliche Routen pro Produkt und somit 1024^{30} mögliche Routenkombinationen aller 30 Produkte. Die Komplexität sowie die Wichtigkeit der optimalen Koordination der MP zur Erreichung einer beispielsweise maximalen Kapazitätsauslastung deuten sich bereits hier an. Den Details der Aufgaben der Produktionssteuerung, welche beispielsweise auch die Koordination der Fahrzeugflotte und den Umgang mit Störungen mit einschließt, widmet sich Abschnitt 2.2. Zunächst geht Abschnitt 2.1.3 auf die logistischen Zielgrößen und Kennzahlen der MP ein, an denen eine Produktionssteuerung gemessen werden kann.

2.1.3 Logistische Zielgrößen und Kennzahlen

Ein Produktionssystem unterliegt wirtschaftlichen Zielgrößen. Wie sich in Abschnitt 2.2 zeigen wird, spielt zur Erreichung dieser Zielgrößen die Produktionssteuerung eine wesentliche Rolle. Nach Wiendahl [38] basiert das zugrunde liegende Zielsystem hoher Wirtschaftlichkeit auf der Erreichung einer möglichst hohen Logistikleistung bei möglichst geringen Logistikkosten, dargestellt in Abbildung 2.3a. Eine hohe Logistikleistung zeigt sich in kurzen Lieferzeiten durch kurze Durchlaufzeiten in der Produktion sowie in einer hohen Liefertreue durch eine zeitige Abarbeitung eingehender Produktionsaufträge und damit verbundene Termintreue. Die Logistikkosten hingegen beziehen sich auf Prozesskosten und Kapitalbindungskosten. Geringe Logistikkosten erfordern somit hohe Stationsauslastungen und niedrige Bestände [38, S. 250ff.].

Hierbei tritt zwischen den Zielen hoher Auslastung, niedriger Bestände, niedriger Durchlaufzeiten und hoher Termintreue ein Zielkonflikt auf, der seit langem als Dilemma der Ablaufplanung bekannt ist [73, S. 158-161]. Hohe Auslastung erfordert hohe Bestände, was wiederum zu hohen Durchlaufzeiten führt. Lange Durchlaufzeiten widersprechen jedoch dem Ziel der hohen Terminsicherheit. Je nach Lagerstrategie lässt sich bestimmen, welche Zielgrößen im Vordergrund stehen sollten. Fertigen Produktionssysteme auf Lager, werden vorwiegend die wirtschaftlichen Ziele der hohen Auslastung und geringer Bestände verfolgt. Durchlaufzeiten und Termintreue spielen hier eine sekundäre Rolle. Bei auftragsbezogenen Produktionssystemen kehrt sich dieses Verhältnis ins Gegenteil um [72, S. 4f.].

Aufgrund dieses Zielkonfliktes und der Abhängigkeiten zwischen Ziel- und Einflussgrößen kommt der Modellierung der Zielgrößen und Abhängigkeiten eine wesentliche Bedeutung zu. Sie dient erstens der späteren Einstellung der Steuerungsparameter in Abstimmung mit den Zielgrößen und zweitens dem Erkenntnisgewinn über die Gestaltung der Produktionssteuerung [74, S. 43]. Die Modellierung folgt meist dem Hannoverschen Trichtermodell, welches in Abbildung 2.3b dargestellt ist. Das Trichtermodell geht davon aus, dass sich jeder Bestandteil des Produktionssystems als Trichter darstellen lässt. Dies trifft genauso auf einzelne Stationen wie auch auf die Modellierung des gesamten Produktionssystems zu [72, S. 25]. Von

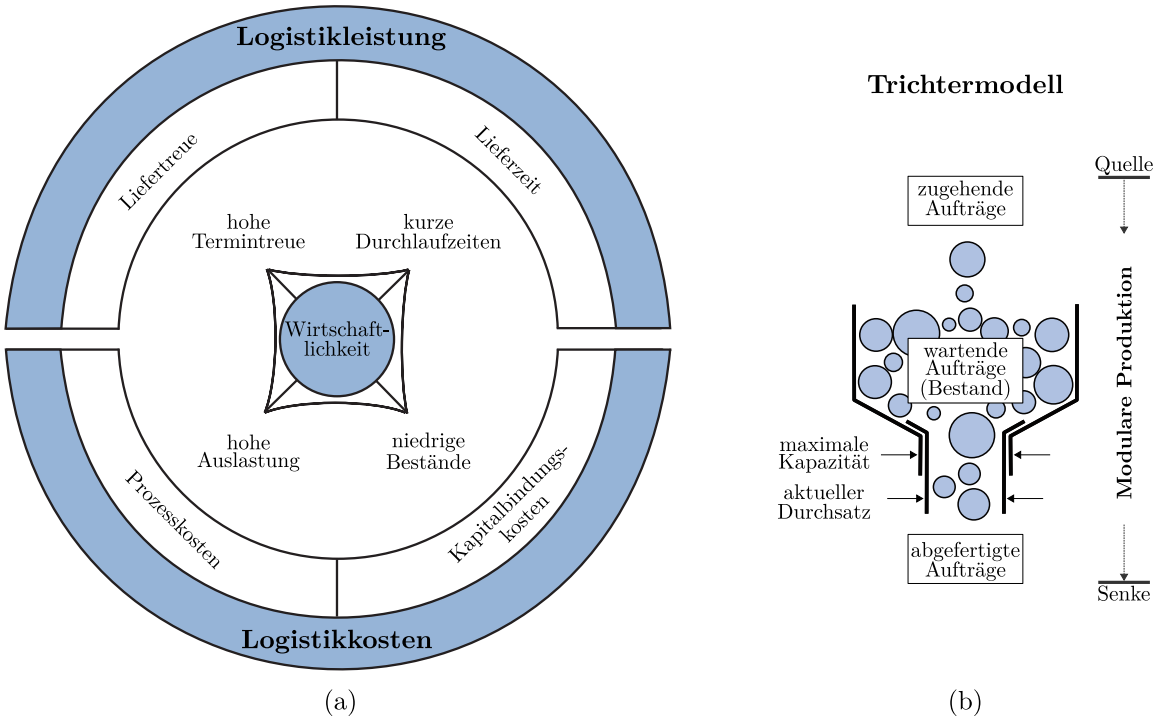


Abbildung 2.3: Zielsystem der Produktionslogistik (a) und Trichtermodell zur logistischen Modellierung der Produktionsprozesse (b), Quelle: Eigene Darstellung in Anlehnung an Wiendahl [38, S. 251] sowie Nyhuis und Wiendahl [72, S. 25].

Nyhuis und Wiendahl [72, S. 25] übertragen auf die MP gilt: Die Produktionsaufträge, die sich bereits im Produktionssystem befinden und ihrem Produktgraphen zur Fertigstellung folgen, gelten zusammen mit den zur Einlastung an der Quelle bereitstehenden Aufträgen als wartende Aufträge und bilden somit den Produktionsbestand PB der MP[3]. PB_m bezeichnet den durchschnittlichen Bestand wartender Aufträge. Der Durchmesser der in Abbildung 2.3b dargestellten Kreise steht hierbei für den jeweiligen Kapazitätsbedarf der Aufträge [74, S. 60]. Ein fertiggestelltes Produkt fließt über die Trichteröffnung in die Senke der MP ab, wobei die Trichteröffnung den Durchsatz des Produktionssystems darstellt [39, S. 254]. Der mittlere Durchsatz D_m der Produktion ist damit als Anzahl fertiggestellter Produktionsaufträge je Zeiteinheit definiert [72, S. 112]:

$$D_m = \frac{N_{\mathrm{AB}}}{t_{\mathrm{BZ}}}, \tag{2.1}$$

wobei N_{AB} die Anzahl der Abgänge im Bezugszeitraum und t_{BZ} die Länge des Bezugszeitraums ist. Je höher der Durchsatz, desto leistungsfähiger ist das Produktionssystem, da es

[3]Zusätzlich zum Produktionsbestand führt Lödding den Begriff des Lagerbestands ein. Dieser beinhaltet zur Produktfertigstellung notwendige Rohmaterialien, Halbfabrikate und Fertigwaren [74, S. 36]. Für die Entwicklung der Steuerungsalgorithmen dieser Arbeit hat der Lagerbestand aufgrund der in Abschnitt 2.4.2 abgeleiteten Annahmen keine Relevanz. Dieser Abschnitt beschränkt sich deshalb auf Produktionsbestand und verwendet die Begriffe Produktionsbestand und Bestand in der Folge als Synonyme.

mehr Produkte in kürzerer Zeit produziert, wobei mehr von der verfügbaren Produktionskapazität genutzt wird. Hierbei repräsentiert die maximale Breite der Trichteröffnung in Abbildung 2.3b die verfügbare Kapazität. Der maximal erreichbare Durchsatz D_{max} ist damit von der vorhandenen Kapazität sowie von den zu fertigenden Produkten abhängig. Die mittlere Auslastung A_m des Produktionssystems setzt den mittleren Durchsatz D_m mit dem maximal möglichen D_{max} ins Verhältnis und berechnet sich damit zu [74, S. 39]:

$$A_m = \frac{D_m}{D_{max}}. \tag{2.2}$$

Die mittlere Auslastung A_m kann sich nach dem Trichtermodell auf den mittleren Wert für das gesamte Produktionssystem oder lediglich auf eine Einzelstation beziehen. Dass der erreichte mittlere Produktionsdurchsatz D_m meist nicht dem theoretischen Maximum entspricht, kann verschiedene Gründe haben. Einer dieser Gründe sind auftretende Stationsstörungen. Für eine möglichst effiziente Nutzung der vorhandenen Produktionskapazität ist diese Dynamik vor allem durch die Steuerung des Produktionssystems zu berücksichtigen [75]. Störungen lassen sich anhand der beiden Kennzahlen „Mean Time Between Failure“ (MTBF) und „Mean Time To Restoration“ (MTTR) klassifizieren. MTBF gibt an, wie viel Zeit im Mittel zwischen zwei Störungen vergeht [76, S. 6]. MTTR hingegen bezeichnet die durchschnittliche Zeit für die Beseitigung der Störung und die damit verbundene Instandsetzung [77, S. 116]. Die Verfügbarkeit V_M einer Arbeitsstation gibt die Wahrscheinlichkeit an, zu einem gegebenen Zeitpunkt ihre Arbeit verrichten zu können und korreliert damit mit MTBF und MTTR [77, S. 119ff.].

Die ebenfalls auf dem Trichtermodell basierende Berechnung der Durchlaufzeit DZ_{J_i} eines Produktionsauftrags J_i in (2.3) bezieht sich auf die Gesamtzeit, die sich ein Auftrag im System befindet. Sie berechnet sich wie folgt [74, S. 32]:

$$DZ_{J_i} = E_{J_i} - S_{J_i}, \tag{2.3}$$

wobei E_{J_i} den Zeitpunkt der Fertigstellung und S_{J_i} den Zeitpunkt der Einlastung von Auftrag J_i bezeichnet. Die gesamte Auftragsdurchlaufzeit setzt sich wiederum aus den Durchlaufzeiten DZ_{ipl} der einzeln durchzuführenden Produktionsvorgänge O_{ipl} an den Stationen einschließlich der Übergangszeiten zwischen den Stationen zusammen, was (2.4) zusammenfasst [72, S. 22]:

$$DZ_{ipl} = E_{ipl} - E_{ip(l-1)} = UEZ_{ipl} + DFZ_{ipl}, \tag{2.4}$$

wobei E_{ipl} das Bearbeitungsende von Vorgang O_{ipl} und $E_{ip(l-1)}$ das Bearbeitungsende vom Vorgänger-Vorgang $O_{ip(l-1)}$ bezeichnet. UEZ_{ipl} ist die Übergangszeit von Vorgang O_{ipl}. Sie setzt sich aus der Liegezeit nach der letzten Bearbeitung, der Transportzeit zur nächsten Sta-

tion und der Liegezeit vor der nächsten Bearbeitung von Vorgang O_{ipl} zusammen. Die Durchführungszeit DFZ_{ipl} bezieht hingegen die für die Bearbeitung notwendige Rüstzeit und die reine Bearbeitungszeit von Vorgang O_{ipl} mit ein. Die durchschnittlichen Auftragsdurchlauf- und Übergangszeiten mehrerer Aufträge werden als $DZ_{J,m}$ und $UEZ_{J,m}$ bezeichnet.

Zum Verständnis der zu Beginn dieses Abschnitts eingeführten Zielkonflikte zwischen den logistischen Kenngrößen lassen sich Produktionskennlinien darstellen. Diese eignen sich für eine analytische Untersuchung unterschiedlicher Betriebszustände mit dem Ziel, eine bestmögliche Zielerfüllung aus Unternehmenssicht zu erreichen. Abbildung 2.4 zeigt beispielhafte Kennlinien in Abhängigkeit vom mittleren Produktionsbestand PB_m. Dargestellt ist der maximal zu erreichende Produktionsdurchsatz D_{max}, der sich an der vorhandenen Kapazität orientiert. Der erreichte mittlere Durchsatz D_m wächst mit steigenden Beständen, nähert sich D_{max} dabei an, erreicht das Maximum jedoch aufgrund der Störfaktoren nicht vollständig. Die mittlere Auftragsdurchlaufzeit $DZ_{J,m}$ und Übergangszeit $UEZ_{J,m}$ steigen hingegen ab einer bestimmten Bestandsgrenze proportional zum Bestand an [38, S. 263f.]. Diese lineare Korrelation zwischen PB_m und $DZ_{J,m}$ bzw. der anteiligen $UEZ_{J,m}$ wird im sogenannten „Gesetz von Little“ beschrieben bzw. davon abgeleitet [78].

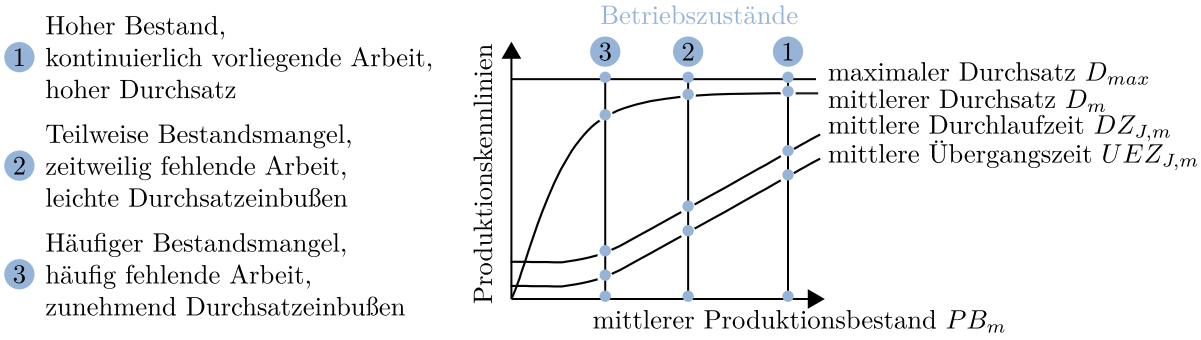


Abbildung 2.4: Produktionskennlinien für Durchsatz, Durchlaufzeit und Übergangszeit in Abhängigkeit des Produktionsbestands zur Darstellung logistischer Zielkonflikte anhand verschiedener Betriebszustände, Quelle: Eigene Darstellung in Anlehnung an Wiendahl [38, S. 264].

Das Gesetz von Little stammt aus der Warteschlangentheorie und stellt einen alternativen, jedoch dem Trichtermodell ähnlichen logistischen Modellierungsansatz dar [72, S. 32]. Es setzt den mittleren Bestand PB_m mit der mittleren Durchlaufzeit $DZ_{J,m}$ mithilfe einer mittleren Ankunftsrate $\lambda_{J,m}$ in Relation. Wird $\lambda_{J,m}$ durch den mittleren Durchsatz D_m, also die Abfertigungsrate als Gegenstück zur Ankunftsrate, ersetzt [72, S. 34], lässt sich der Zusammenhang wie in (2.5) aus Sicht des Durchsatzes beschreiben:

$$D_m = \frac{PB_m}{DZ_{J,m}}, \tag{2.5}$$

wobei PB_m der mittlere Produktionsbestand und $DZ_{J,m}$ die mittlere Durchlaufzeit der Menge der Aufträge J ist. Bei einem gegebenen maximalen Produktionsbestand PB_{max}, weil

beispielsweise nur eine bestimmte Anzahl an WTs zur Verfügung steht, lässt sich der Durchsatz folglich durch eine Minimierung der Durchlaufzeiten maximieren. Dieser Zusammenhang gilt nicht unbegrenzt, weil erstens Durchlaufzeiten mit dem Produktionsbestand korrelieren und zweitens die Durchlaufzeiten nicht unbegrenzt verringert werden können. Jedoch gibt der Zusammenhang Anhaltspunkte, wie sich der Durchsatz einer MP hin zu D_{max} optimieren lässt. Außerdem kann der Zusammenhang zur Auslegung eines für die Erreichung von D_{max} mindestens notwendigen Produktionsbestandes PB_{min} eingesetzt werden. Sind der erreichbare Durchsatz D_{max} sowie die minimal erreichbaren Durchlaufzeiten $DZ_{J,m,min}$ der zu fertigenden Produkte bekannt, kann auf den mindestens notwendigen Produktionsbestand $PB_{min} \geq D_{max} \cdot DZ_{J,m,min}$ geschlossen werden.

Weitere Herleitungen und eine ausführliche Darstellung des Trichtermodells sowie des Gesetzes von Little finden sich in der Arbeit von Nyhuis und Wiendahl [72, S. 17-38]. Wie die logistischen Kenngrößen durch eine Produktionssteuerung optimiert werden können, thematisiert der nächste Abschnitt.

2.2 Steuerung der Modularen Produktion

Lödding [74, S. 1ff.] bezeichnet die Produktionssteuerung als einen wesentlichen Erfolgsfaktor produzierender Unternehmen, logistische Ziele wie geringe Bestände sowie pünktliche und schnelle Lieferungen zu erreichen, welche in Abschnitt 2.1.3 eingeführt wurden. Laut Lödding offenbart die Praxis jedoch erhebliche Verbesserungspotenziale in der logistischen Zielerreichung, weil sich einige Unternehmen zum Beispiel zu stark auf von der Produktionsplanung berechnete Ablaufpläne verlassen und zu wenig Wert auf die Umsetzung dieser Pläne durch eine funktionierende Steuerung legen [74, S. 1ff.]. Um dem in Bezug auf die MP entgegenzuwirken, legt diese Arbeit den Fokus auf eine funktionierende Steuerung, die das Erreichen der logistischen Ziele anstrebt. Demzufolge beschreibt Abschnitt 2.2.1 zuerst Grundlagen zur Produktionsplanung und -steuerung (PPS), bevor Abschnitt 2.2.2 die zu treffenden Entscheidungen in der MP einführt. Anschließend stellt Abschnitt 2.2.3 Scheduling als Entscheidungsprozess für die vorgestellten Entscheidungskategorien dar. Abschnitt 2.2.4 beschreibt abschließend verschiedene Steuerungsparadigmen, die im Scheduling zur Modellierung der Lösungsalgorithmen Anwendung finden können.

2.2.1 Produktionsplanung und -steuerung

Die PPS hat nach Wiendahl [38, S. 249] zwei zentrale Aufgaben: die regelmäßige Vorausplanung des laufenden Produktionsprogramms für mehrere Planungsperioden (Planung) und die möglichst wirtschaftliche Realisierung dieses Programms trotz auftretender Störfaktoren wie Maschinenausfallzeiten oder Lieferverzögerungen (Steuerung).

Zur Beschreibung dieser zwei zentralen Aufgaben wird häufig das allgemeingültige Aachener PPS-Modell [79, S. 16f.] als Referenz verwendet, welches sich im Wesentlichen auf zwei Modelle bezieht: das 1989 von Hackstein [80, S. 5] entwickelte PPS-Modell und das 1995 von Scheer [81, S. 93] entwickelte Y-Modell. Hacksteins Darstellung erwies sich als treffend, um die in dieser Arbeit betrachtete Problemstellung prägnant darzustellen.

Basierend auf Hacksteins Arbeit zeigt Abbildung 2.5 die Funktionsgruppen der beiden PPS-Teilgebiete: Planung und Steuerung. Die Planung umfasst die Produktionsprogrammplanung, die Materialbedarfsplanung sowie die Termin- und Kapazitätsplanung. Die Steuerung hingegen umfasst die Auftragsfreigabe und die Auftragsüberwachung. Die Datenverwaltung stellt eine Querschnittsfunktion der PPS dar, da eine funktionierende Datenverwaltung für die Speicherung und Verarbeitung von Grund-, Kundenauftrags- sowie fertigungsspezifischen Daten notwendig ist und alle anderen Funktionsgruppen unterstützt [82, S. 72],[83, S. 13].

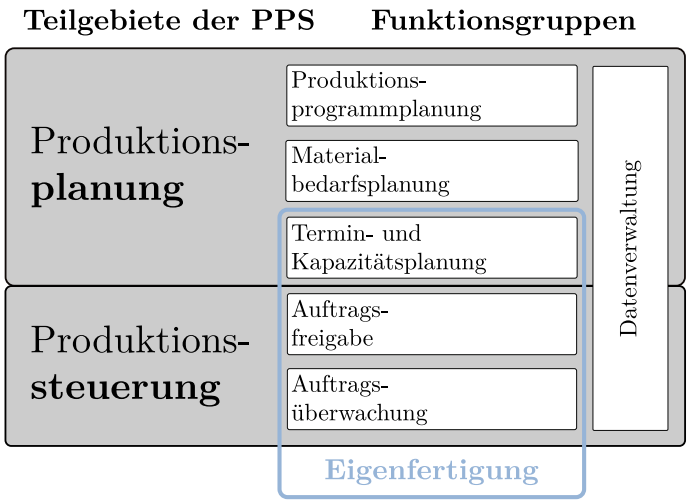


Abbildung 2.5: Teilgebiete der PPS und ihre Funktionsgruppen, Quelle: Eigene Darstellung in Anlehnung an Wiendahl [38, S. 282] und Hackstein [80, S. 5].

Produktionsplanung: Die z.B. monatlich rollierende Produktionsprogrammplanung definiert Art, Menge und Termin der zu produzierenden Erzeugnisse für einen festgelegten Zeitraum. Der resultierende Produktionsplan berücksichtigt dabei die geplanten Absatzzahlen und wird in enger Abstimmung von Produktion und Vertrieb erarbeitet [82, S. 39], [80, S. 89ff.]. Auf Basis des Produktionsprogramms leitet die beinahe täglich stattfindende Materialbedarfsplanung die Bedarfe an Eigenfertigungs- und Zukaufteilen ab. Insbesondere werden Beschaffung, Lager, Entsorgung und Bestände organisiert, bevor die Disposition Fertigungsaufträge für die Eigenfertigung und Bestellaufträge für Zukaufteile erzeugt. Für Eigenfertigungsteile folgt die Termin- und Kapazitätsplanung, welche den Fertigungsaufträgen Starttermine zuweist, die Kapazitätsbedarfe mit dem Kapazitätsangebot abgleicht und die Reihenfolgeplanung der Aufträge an den Kapazitätseinheiten übernimmt [38, S. 282f.].

Produktionssteuerung: Produktionssteuerungsaufgaben werden sowohl für die Eigenfertigung als auch für die Bestellaufträge benötigt. Da sich diese Arbeit mit der Eigenfertigung in einer MP befasst, beschränkt sie sich darauf. Die Auftragsfreigabe und die Auftragsüberwachung bilden die Funktionen der Produktionssteuerung. Nach Eversheim [40, S. 150] erfolgt die Auftragsfreigabe, nachdem die Verfügbarkeit der Betriebsmittel, des Personals, des Materials und der Informationen geprüft wurde. Sobald ein Auftrag freigegeben wurde, kann er in das Produktionssystem eingelastet werden. Für die Festlegung des tatsächlichen Einlastungszeitpunkts existieren nach Lödding [74, S. 339f.] vier Mechanismen bzw. Kriterien: sofortige Auftragsfreigabe, Plan-Starttermin, Produktionsbestand und Belastung der Produktion, beschrieben in Tabelle 2.2. Den Moment der Einlastung bezeichnet diese Arbeit als Auftragseinlastung[4].

Tabelle 2.2: Kriterien für Einlastungszeitpunkt nach Lödding [74, S. 339f.].

Kriterium	**Beschreibung**
Sofortige Auftragsfreigabe	Einlastung des Auftrags in das Produktionssystem zum Zeitpunkt der Erzeugung.
Plan-Starttermin	Einlastung des Auftrags zu einem vordefinierten Termin.
Bestand der Produktion	Regelung der Einlastungszeitpunkte durch den Bestand der Produktion, beispielsweise bei Unterschreitung eines Planwerts.
Belastung der Produktion	Regelung der Einlastungszeitpunkte entsprechend der momentanen oder erwarteten Belastung der Ressourcen, basierend auf Informationen über den Bestand des Systems.

Lödding beschreibt in [74, S. 339-503] einige Verfahren, die sich den Kriterien in Tabelle 2.2 zuordnen lassen. Die ursprünglich von Bechte [84] entwickelte und von Wiendahl [85, S. 284ff.] aufgegriffene belastungsorientierte Auftragsfreigabe (BOA) zählt zu den bekanntesten Verfahren. Die BOA regelt das System über den Bestand der Produktion. Aufträge aus einer vorgegebenen Liste werden sequenziell dann eingelastet, wenn keine der Maschinen bzw. Stationen eine vordefinierte Bestandsgrenze überschreitet. Dies ermöglicht einen Belastungsabgleich bei der Auftragseinlastung und verhindert eine Überlastung des Systems. Ein weiteres weit verbreitetes Einlastungsverfahren ist Conwip, was für „Constant Work in Process“ steht und von Spearman und Hopp geprägt wurde [86]. Der Name verrät die Grundidee des Verfahrens, den Bestand der Produktion konstant zu halten. Demnach startet Conwip einen neuen Produktionsauftrag, wenn eine Bestandsgrenze PB_{max} des gesamten Produktionssystems unterschritten wird. In der MP ist diese Bestandsgrenze beispielsweise der Anzahl der verfügbaren WTs gleichzusetzen. Beispiele für weitere Auftragsfreigabe-Verfahren sind

[4] Lödding [74] bezeichnet die Verfahren, obwohl sie den Zeitpunkt der Einlastung definieren, als Auftragsfreigabe-Verfahren.

die „Workload Control“ und die „Polca-Steuerung“ [74, S. 407ff., S. 467ff.].

Nach der Auftragseinlastung beginnt die Auftragsüberwachung. Diese überwacht den Auftragsfortschritt auf Basis ständiger Rückmeldungen aus der Produktion, welche beispielsweise über ein System zur Betriebsdatenerfassung (BDE) gesammelt werden [38, S. 283]. Eine direkte Einflussnahme auf bereits eingelastete Aufträge ist in der klassischen PPS allerdings nicht vorgesehen [80, S. 223-244, S. 249].

Da zur Nutzung der Flexibilität der MP die aktive Einflussnahme auf bereits eingelastete Aufträge von elementarer Bedeutung ist, befassen sich die nächsten Abschnitte mit einer Erweiterung des hier vorgestellten Modells.

2.2.2 Entscheidungskategorien zur Steuerung der Modularen Produktion

In der in Abschnitt 2.1 vorgestellten MP können Produktionsaufträge flexibel abgearbeitet werden. Beispielsweise besteht die Auswahlmöglichkeit zwischen alternativen Routen, die durch den flexiblen Materialtransport durch FTFs dynamisch angepasst werden können. Im Vergleich zu einer konventionellen Linienfertigung ist deshalb eine direkte Einflussnahme auf eingelastete Aufträge erforderlich, um beispielsweise im Störungsfall Aufträge umzuleiten. Die klassische PPS sieht diese Einflussnahme nicht vor. Zur Ermöglichung der Einflussnahme sind die in der MP zu treffenden Entscheidungen zuerst zu definieren.

Ergänzend zu Wiendahl [38, S. 282] und Hackstein [80, S. 5] teilt Baker [87] die Produktionssteuerung in Low- und Highlevel-Steuerungsaufgaben ein. Auf einem niedrigeren Level werden im Wesentlichen die Ressourcen der Produktion gesteuert, wobei beispielsweise eine speicherprogrammierbare Steuerung (SPS) die Fertigungsprozesse einer Maschine verantwortet, was nicht Teil dieser Arbeit ist. Auf einem höheren Level erfolgt hingegen die Koordination der Ressourcen, um die zu fertigenden Produkte möglichst wirtschaftlich zu realisieren, analog zur PPS. Leitao [88] greift hiervon einige Entscheidungen auf. Zu den Steuerungsaufgaben zählt er, wann welche Produktionsaufträge in die Produktion eingelastet werden, auf welcher Route die Aufträge durch das System geleitet werden und wie einzelne Operationen konkurrierender Aufträge an den Maschinen sequenziert werden.

Auf die MP bezogen erweitern [29], [32] und [33] diese Steuerungsentscheidungen um ein Flottenmanagement zur Koordination einer FTF-Flotte und um ein Störungsmanagement zum Umgang mit Störfaktoren wie Maschinenausfällen. Zusammen mit den in Abschnitt 2.2.1 vorgestellten Steuerungsaufgaben lassen sich für bereits freigegebene Produktionsaufträge in der MP fünf Entscheidungskategorien ableiten, welche in Abbildung 2.6 visualisiert werden. Die vorliegende Arbeit bezeichnet sie als Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement nach folgender Definition:

- **Auftragseinlastung**: Die Auftragseinlastung definiert, wann welcher Produktionsauftrag aus dem Produktionsprogramm in das System eingelastet wird.
- **Stationsmanagement**: Das Stationsmanagement sequenziert die um eine Maschine konkurrierenden Produktionsaufträge.
- **Auftragsrouting**: Das Auftragsrouting bestimmt, welche Route ein Produktionsauftrag auf Basis seiner Reihenfolge- und Operationsflexibilität durch das System wählt.
- **Flottenmanagement**: Das Flottenmanagement weist jeden Transportauftrag einem FTF der Flotte zu und koordiniert die Transporte unter Vermeidung von Deadlocks.
- **Störungsmanagement**: Das Störungsmanagement reagiert auf Störfaktoren wie Maschinenausfälle und steuert die Ressourcen und Aufträge entsprechend.

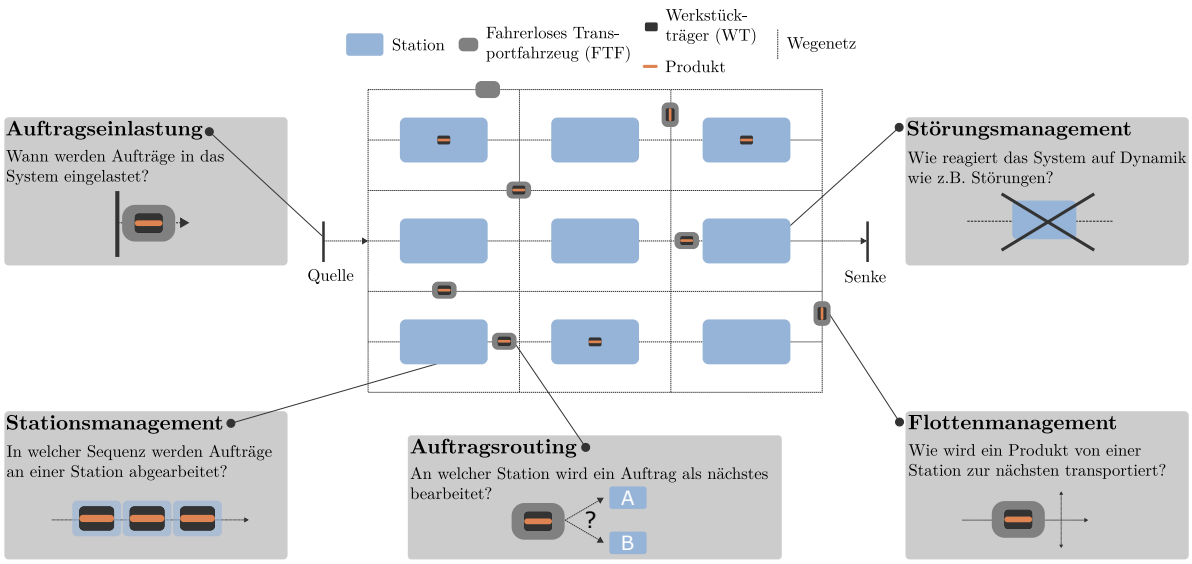


Abbildung 2.6: Entscheidungskategorien für die Produktionssteuerung in einer MP, Quelle: Eigene Darstellung in Anlehnung an [29], [32], [33].

Damit die MP mit ihrer Steuerung in den fünf Entscheidungskategorien funktionieren kann, befasst sich die Produktionslogistik mit logistischen Tätigkeiten, Maßnahmen und Themenstellungen für die Waren bzw. Leistungserbringung in der Produktion. Hierzu zählen auch Planung, Steuerung, Transport und Lagerung von Rohmaterialien, Hilfs- sowie Betriebsstoffen, die zur Bearbeitung der Produkte an Stationen benötigt werden [89, S. 238]. Die Arbeit geht an dieser Stelle davon aus, dass die Bereitstellung der benötigten Materialien analog dem Konzept von Kern et al. [64] über dezentrale Lagerflächen direkt an den Stationen erfolgt und unabhängig von der Produktionssteuerung geregelt wird. Folglich befasst sich die Steuerung der MP durch das Flottenmanagement lediglich mit den Logistiktransporten der Produkte von Station zu Station.

Die Kenntnis der in diesem Abschnitt vorgestellten fünf Entscheidungskategorien stellt für die MP sicher, dass nach erfolgter Auftragseinlastung die direkte Einflussnahme auf die eingelasteten Produktionsaufträge erfolgen kann. Außerdem stellen die Entscheidungskategorien wesentliche Anforderungen an die in dieser Arbeit entwickelten Steuerungsalgorithmen. Zur Entscheidungsfindung in den Kategorien führt der folgende Abschnitt Scheduling ein.

2.2.3 Scheduling als Entscheidungsprozess

Entscheidungen in den in Abschnitt 2.2.2 vorgestellten Entscheidungskategorien der MP werden in verwandten Produktionsarten im Zusammenhang mit sogenannten Scheduling-Problemen[5] behandelt, wobei die Begriffe Ablaufplanung und Maschinenbelegungsplanung meist als Synonyme für Scheduling verwendet werden. Diese Arbeit legt sich auf den Begriff Scheduling fest, der vor allem in der englischsprachigen Literatur zu finden ist. Pinedo [90, S. 1] definiert Scheduling als regelmäßig stattfindenden Entscheidungsprozess, der im Produktions- und Dienstleistungsbereich Anwendung findet. Im Produktionsbereich schließt Scheduling nach Watermeyer [70, S. 9f.] an den konventionellen PPS-Prozess an und folgt damit der Auftragsfreigabe.

Scheduling hat im Produktionsbereich die Aufgabe, für eine gegebene Zeitspanne durchzuführende Produktionsvorgänge zu Ressourcen optimal zuzuweisen, wobei ein oder mehrere Zielkriterien wie z.B. die Produktionsspanne oder die Durchlaufzeiten aller Aufträge zu optimieren sind [91]. Es handelt sich um ein kombinatorisches Optimierungsproblem [92]. Hierbei zählen Scheduling-Probleme zu den komplexesten kombinatorischen Optimierungsproblemen und werden in Abhängigkeit der Problemgröße i.d.R. als NP-vollständig[6] eingestuft [94], [95]. Die optimierte Ressourcenallokation wird nach Pinedo [90, S. 21-25] als Schedule bezeichnet und kann in sogenannten Gantt-Diagrammen visualisiert werden. Das Gantt-Diagramm stellt dabei die entschiedene Allokation durch Balken bezogen auf die Zeitachse dar, wie z.B. in [96]. Diese Arbeit verwendet die Begriffe Schedule und Gantt-Diagramm synonym.

Abbildung 2.7 zeigt einen Beispielschedule. Sie stellt die Allokation von fünf Produktionsaufträgen (Farbstufen) aus drei Produkttypen (Farbtypen) zu drei Stationen $M := \{M_1, M_2, M_3\}$ dar. Die Transporte erfolgen mittels fünf FTFs $V := \{V_1, V_2, V_3, V_4, V_5\}$. Die Länge der Balken repräsentiert damit entweder die Bearbeitungszeit an einer Station oder die Transportzeit auf einem FTF. Die Beendigung der letzten Bearbeitung markiert die Produktionsspanne aller Aufträge, also die Zeit der Abarbeitung aller Aufträge.

Schedules können sowohl statisch als auch dynamisch berechnet werden. Ein statischer Schedule, beispielsweise für einen ganzen Produktionstag, optimiert die Ressourcenbelegung

[5] Scheduling-Probleme in Produktionssystemen existieren mit unterschiedlichen Flexibilitäts- und Komplexitätsstufen. Näheres hierzu ist beispielsweise in Pinedo [90], Watermeyer [70] oder Ouelhadj und Petrovic [75] nachzulesen.

[6] In NP-vollständig steht NP für „nicht-deterministische Polynomialzeit“ und beschreibt eine Gruppe komplexer Optimierungsprobleme. Derartige Probleme sind nicht effizient lösbar [93, S. 53ff.].

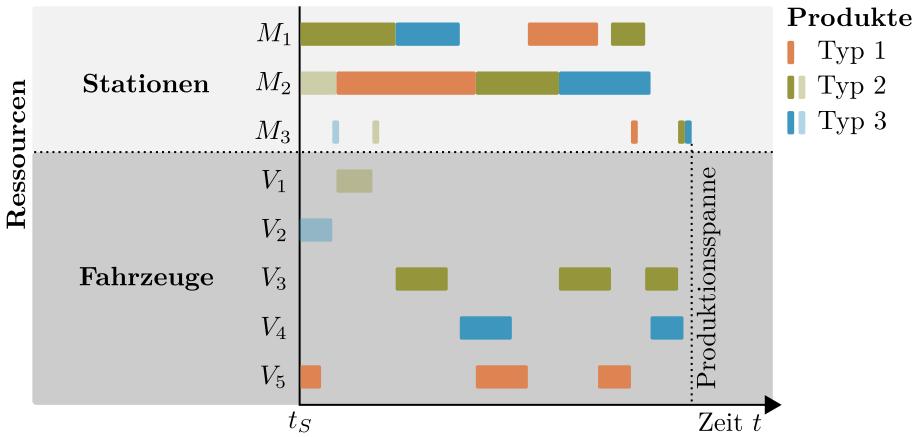


Abbildung 2.7: Darstellung der Allokation von fünf Produktionsaufträgen (Farbstufen) aus drei Produkttypen (Farbtypen) zu drei Stationen $M := \{M_1, M_2, M_3\}$ und fünf FTFs $V := \{V_1, V_2, V_3, V_4, V_5\}$.

nach bestimmten Kriterien a priori für alle vorliegenden Aufträge und geht davon aus, dass die nötigen Informationen zu allen an diesem Tag zu fertigenden Produktionsaufträgen deterministisch vorliegen. Störungen werden vorerst nicht berücksichtigt [75].

Da Produktionssysteme i.d.R. in dynamischer Umgebung agieren und unvorhersehbare Störfaktoren wie Maschinenausfälle Änderungen an statischen Schedules erfordern, entsteht eine Lücke zwischen Lösungen der statischen Scheduling-Theorie und den Abläufen in realen, dynamischen Produktionssystemen [97], [98]. Für die reale MP der vorliegenden Arbeit liegt der Fokus auf dynamischem Scheduling. Der Entscheidungskategorie Störungsmanagement kommt daher eine wesentliche Bedeutung in der Produktionssteuerung zu.

Ouelhadj und Petrovic [75] differenzieren in dynamischen Produktionsumgebungen zwei Arten von Störfaktoren, die die Lücke zwischen der statischen und dynamischen Theorie begünstigen: auftragsbezogene und ressourcenbezogene Ereignisse. Zu den auftragsbezogenen Ereignissen zählen z.B. Eilaufträge oder Auftragsstornierungen, während zu den ressourcenbezogenen beispielsweise Maschinenausfälle und zur Bearbeitung fehlende Werkzeuge zählen. Zum Umgang mit derartigen Ereignissen klassifizieren Ouelhadj und Petrovic primär drei Lösungsfelder: vollständig reaktives Scheduling, prädiktiv-reaktives Scheduling und robust-proaktives Scheduling. Hinzu kommt eine Möglichkeit des rollierenden Schedulings, wo statische Pläne in definierten Zeitabständen neu berechnet werden. Als Alternative zu den eben genannten eher zentral organisierten Scheduling-Ansätzen findet das dezentral organisierte Multiagenten-Scheduling immer häufiger Anwendung, welches aufgrund der verteilten Architektur mehr Robustheit im Umgang mit Echtzeitereignissen verspricht [75]. Auch Mischformen der genannten Ansätze finden Anwendung. Tabelle 2.3 beschreibt die wesentlichen Eigenschaften der fünf genannten Lösungsfelder und fasst dabei die Ausführungen in [75], [99]–[101] zusammen. Die Analyse alternativer Ansätze aus den fünf Feldern erfolgt in den späteren Kapiteln zur Entwicklung der Steuerungsalgorithmen.

Tabelle 2.3: Lösungsfelder des dynamischen Schedulings nach [75], [99]–[101].

Lösungsfeld	Beschreibung
Reaktiv	Reaktives Scheduling arbeitet ohne einen vorher berechneten globalen statischen Schedule. Entscheidungen werden lokal und in Echtzeit getroffen, häufig mithilfe zentral verwalteter Prioritätsregeln. Prioritätsregeln wie „Shortest Queue“ (SQ), wobei ein Auftrag der Maschine mit der kürzesten Warteschlange zugewiesen wird [90, S. 376], agieren kurzsichtig und bieten daher in den meisten Fällen eine eingeschränkte Lösungsgüte.
Prädiktiv-reaktiv	Prädiktiv-reaktives Scheduling ist der am häufigsten verwendete dynamische Ansatz. Global erzeugte statische Schedules (prädiktives Scheduling) werden im Fall von Echtzeitereignissen überarbeitet (reaktives Rescheduling). So kann gesamtheitlich und vorausschauend optimiert werden, mit Vorteilen bei der Lösungsgüte.
Robust-proaktiv	Robust-proaktives Scheduling fokussiert sich auf die Erstellung möglichst robuster prädiktiver Schedules. Dieser Ansatz integriert vorhersehbare Störereignisse durch Einstreuung zusätzlicher Pufferzeiten in den Schedule. Die fehlende Vorhersagbarkeit der Störereignisse ist hierbei die größte Schwierigkeit.
Rollierend	Rollierendes Scheduling zerlegt das dynamische Scheduling-Problem in eine Serie statischer Probleme, indem die Produktionszeit in Zeitfenster eingeteilt wird [90, S. 402-407]. Die Steuerung löst Neuberechnungen entweder ausschließlich zeitlich periodisch oder zusätzlich bei Echtzeitereignissen aus, weshalb rollierende Ansätze auch als prädiktiv-reaktiv interpretiert werden können. Herausfordernd ist die Einstellung des Planungshorizonts.
Multiagenten	Im Multiagenten-Scheduling kollaborieren Agenten zum Treffen von Echtzeitentscheidungen in einer dezentralen Architektur. Ein zentral erstellter globaler Schedule existiert deshalb i.d.R. nicht. Der Umgang mit Störfaktoren erfolgt demnach ebenfalls dezentral durch die betroffenen Agenten, was die Robustheit fördert (vgl. Abschnitt 2.3.2).

Eine wesentliche Eigenschaft der MP ist der Materialtransport durch ein FTS und damit das in Abschnitt 2.2.2 definierte Flottenmanagement. Die klassische Scheduling-Theorie nach Pinedo [90] betrachtet diesen Materialtransport durch autonome Fahrzeuge i.d.R. nicht. Durch den Aufschwung autonomer Fahrzeuge in den letzten Jahren befassten sich trotzdem immer wieder Arbeiten mit FTF-Scheduling. Mousavi et al. [102] sowie Hwang und Kim [103] fokussieren sich beispielsweise auf eine optimale Zuweisung von Transportaufträgen innerhalb einer Fahrzeugflotte. Die Ansätze von Zheng et al. [104] integrieren hingegen die Fahrzeuge als Ressourcen in ein statisches Scheduling-Problem, wohingegen Umar et al. [105] Fahrzeuge in ein dynamisches Scheduling-Problem einbauen. Weitere Veröffentlichungen kon-

zentrieren sich auf die optimale Routenfindung der Fahrzeuge in einem Straßennetz und auf die kollisionsfreie Koordination der Transporte [106], [107]. Hierbei werden kürzeste Pfade meist mit einem A*-Algorithmus gefunden. Der A*-Algorithmus durchsucht das als mathematischer Graph modellierte Transportnetz. Einen Überblick zu diesen Themen verschaffen die Arbeiten von Qiu et al. [108] und Nouri et al. [109].

Abbildung 2.8 ordnet Scheduling als Entscheidungsprozess der MP in das von Hackstein vorgestellte PPS-Modell ein und erweitert es damit um die Einflussnahme auf eingelastete Produktionsaufträge. Wie in [110, S. 262] beschrieben, arbeitet Scheduling hierbei an der Schnittstelle zwischen der Auftragsfreigabe und der tatsächlichen Abarbeitung der Aufträge. Scheduling schließt mit dieser Definition die Entscheidung über die tatsächliche Auftragseinlastung im Gegensatz zur Auftragsfreigabe ein. Im Kontext dieser Arbeit befasst sich Scheduling folglich mit den fünf in Abschnitt 2.2.2 vorgestellten Entscheidungskategorien Auftragseinlastung, Auftragsrouting, Stationsmanagement, Flottenmanagement und Störungsmanagement. Die Auftragsüberwachung bleibt für jeden freigegebenen Auftrag erhalten. Außerdem stellt die Datenverwaltung weiterhin die benötigten fertigungsspezifischen Informationen zur Verfügung.

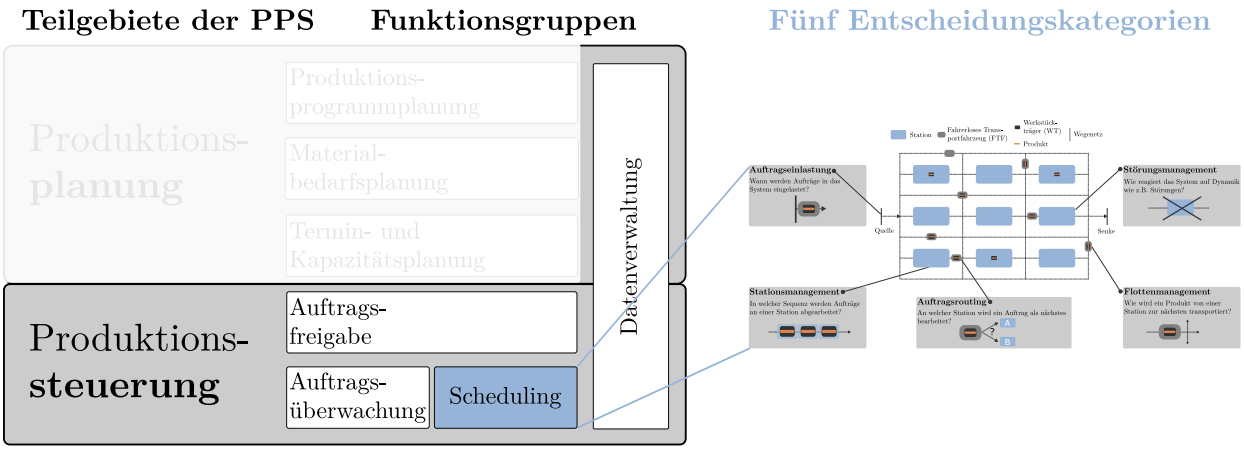


Abbildung 2.8: Erweiterung des PPS-Modells nach Wiendahl [38, S. 282] und Hackstein [80, S. 5] um die Funktion des Schedulings als Entscheidungsprozess für die fünf Entscheidungskategorien Auftragseinlastung, Auftragsrouting, Stationsmanagement, Flottenmanagement und Störungsmanagement, Quelle: Eigene Darstellung in Anlehnung an Abbildung 2.5 und Abbildung 2.6.

Das resultierende Scheduling-Problem für die fünf Entscheidungskategorien der MP kann abschließend, orientiert an der in [70, S. 13-18] verwendeten Notation, wie folgt formalisiert werden: Eine Menge $M := \{M_1, M_2, \ldots, M_k, \ldots, M_{n_M}\}$ von n_M Stationen M_k steht zur Verfügung, um eine Menge $J := \{J_1, J_2, \ldots, J_i, \ldots, J_{n_J}\}$ von n_J Produktionsaufträgen J_i zu produzieren. Der Index k identifiziert eine Station in der Menge M. Der Index i identifiziert einen Auftrag in der Menge J. S_{J_i} bezeichnet den Zeitpunkt der Auftragseinlastung eines Auftrags J_i in der Quelle der MP, E_{J_i} seine Fertigstellung in der Senke der MP. Der Materialtransport zwischen der Quelle, den Stationen und der Senke der MP erfolgt mithilfe einer Menge $V :=$

$\{V_1, V_2, \ldots, V_j, \ldots, V_{n_V}\}$ von n_V FTFs V_j, die sich mit einer einheitlichen Geschwindigkeit v_V auf einem vorgegebenen Straßennetz bewegen. Der Index j identifiziert ein Fahrzeug in der Menge V. Damit die Aufträge bzw. Produkte transportiert werden können, stehen n_{WT} WTs zur Verfügung: $WT := \{WT_1, WT_2, \ldots, WT_o, \ldots, WT_{n_{\mathrm{WT}}}\}$. Ein von einem WT_o getragenes Produkt kann wiederum von Fahrzeugen aufgenommen und von Station zu Station transportiert werden. Die Anzahl der verfügbaren WTs begrenzt den maximal erreichbaren Produktionsbestand zu: $PB_{max} = n_{\mathrm{WT}}$. Produktionsaufträge J_i repräsentieren bestimmte Produkttypen PT_x aus der Menge $PT := \{PT_1, PT_2, \ldots, PT_x, \ldots, PT_{n_{\mathrm{PT}}}\}$. Je nach Produkttyp bietet ein Auftrag ein flexibles Auftragsrouting, welches durch Reihenfolge- oder Operationsflexibilität der Produktionsvorgänge entsteht. Diese Flexibilität steht der Produktionssteuerung als Entscheidungsspielraum zur Verfügung. Die Reihenfolgeflexibilität äußert sich durch n_P Prozesspläne $P_i := \{P_{i1}, P_{i2}, \ldots, P_{ip}, \ldots, P_{in_P}\}$ pro Auftrag J_i, wobei ein Prozessplan P_{ip} aus n_O Produktionsvorgängen bzw. Operationen $O_{ip} := \{O_{ip1}, O_{ip2}, \ldots, O_{ipl}, \ldots, O_{ipn_O}\}$ mit fester Sequenz besteht. Der Index p identifiziert einen Prozessplan in der Menge P_i. Der Index l identifiziert einen Vorgang in der Menge O_{ip}. Die Bearbeitungszeiten von Vorgängen O_{ipl} werden mit $t_{ipl} \in \mathbb{Z}_{\geq 0}$ beschrieben, ihre Startzeiten an den Stationen mit $S_{ipl} \in \mathbb{R}_{\geq 0}$. Operationsflexibilität liegt vor, wenn ein Produktionsvorgang O_{ipl} eines Produktionsauftrags J_i von mindestens zwei Stationen M_k bearbeitet werden kann. Die Menge $M_{ipl} \subseteq M$ als Teilmenge aller Stationen M bezeichnet dann die m_{ipl} alternativen Stationen für Operation O_{ipl}, wobei $m_{ipl} > 1$ gilt (Definition in Anlehnung an [111]). Damit bieten die Stationen in M_{ipl} die notwendigen Fähigkeiten F_{ipl} zur Abarbeitung von O_{ipl} (analog zu Fähigkeitenabgleich in Abschnitt 2.1.2). Zwischen zwei Operationen O_{ipl} und $O_{ip(l+1)}$ vergeht die in Abschnitt 2.1.3 eingeführte Übergangszeit UEZ_{ipl}. Sie setzt sich zusammen aus der Liegezeit im Ausgangspuffer (AP) nach der letzten Bearbeitung $t_{\mathrm{AP},ip(l-1)}$, der Transportzeit zur nächsten Station $t_{\mathrm{TR},ipl}$ und der Liegezeit im Eingangspuffer (EP) vor der nächsten Bearbeitung $t_{\mathrm{EP},ipl}$. Ein Schedule ist in Erweiterung zu Watermeyer [70, S. 14] gültig, wenn er die folgenden Ungleichungen erfüllt:

$$\begin{aligned} & S_{ipl} + t_{ipl} + UEZ_{ipl} \leq S_{ip(l+1)}, \\ & \qquad \forall J_i \in J, \forall P_{ip} \in P_i, \forall O_{ipl} \in O_{ip} \setminus \{O_{ipn_O}\}, \end{aligned} \tag{2.6}$$

$$\begin{aligned} & S_{ipl} + t_{ipl} \leq S_{i'p'l'} \vee S_{i'p'l'} + t_{i'p'l'} \leq S_{ipl}, \\ & \qquad \forall J_i, J_{i'} \in J, i \neq i', \\ & \qquad \forall P_{ip} \in P_i, \forall P_{i'p'} \in P_{i'}, \forall O_{ipl} \in O_{ip}, \forall O_{i'p'l'} \in O_{i'p'}, \\ & \qquad M_{k,ipl} = M_{k,i'p'l'}. \end{aligned} \tag{2.7}$$

Ungleichung (2.6) formuliert, dass Vorgänge eines Auftrags nicht gleichzeitig abgearbeitet werden können. Zwischen zwei Operations-Startzeitpunkten S_{ipl} und $S_{ip(l+1)}$ vergeht die Durchführungszeit DFZ_{ipl} und die Übergangszeit UEZ_{ipl}. Da Rüstzeit in dieser Arbeit ver-

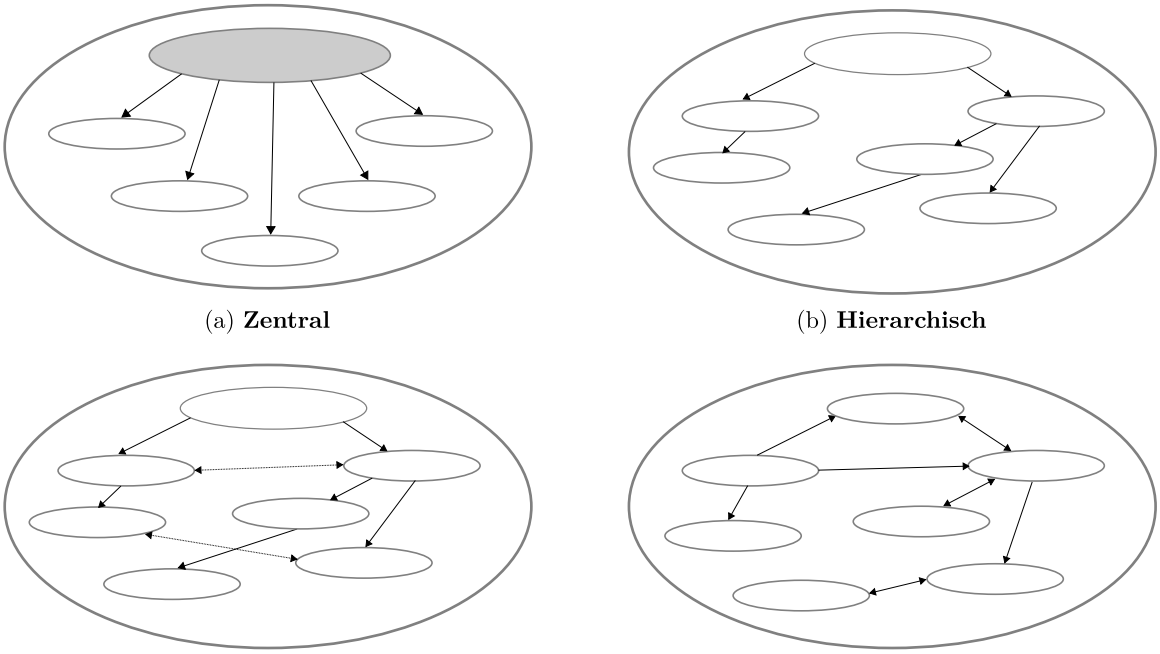
nachlässigt wird, reduziert sich DFZ_{ipl} zur reinen Bearbeitungszeit t_{ipl}. Ungleichung (2.7) beschreibt, dass jede Station nur einen Auftrag nach dem anderen bearbeiten kann, ohne Überschneidungen. Gleiches gilt für die Transporte auf FTFs: Die Transporte eines Auftrags J_i können nicht gleichzeitig durchgeführt werden. Außerdem kann jedes Fahrzeug V_j nur einen Auftrag J_i nach dem anderen transportieren.

Ziel der Steuerung der MP ist die Optimierung der Ressourcenallokation für die fünf Entscheidungskategorien in diesem formal beschriebenen Scheduling-Rahmen. Die Datenverwaltung stellt hierfür die notwendigen Produktionsdaten bereit. Die Optimalität der Scheduling-Lösung bezieht sich auf die in Abschnitt 2.1.3 eingeführten Kenngrößen. In der Scheduling-Literatur findet die Optimierung der Produktionsspanne am häufigsten Anwendung [112]. Da sich die MP in dieser Arbeit weniger auf Kundenauftragsproduktion bezieht, sondern einen bestimmten Produktbestand auf Lager produziert, stehen die wirtschaftlichen Kenngrößen im Vordergrund (vgl. Abschnitt 2.1.3). Die MP strebt in dieser Arbeit eine Maximierung des mittleren Durchsatzes D_m an. Außerdem steht die Robustheit gegenüber Ressourcenstörungen durch das Störungsmanagement im Vordergrund [29], [32], [33]. Die Optimierung der Ressourcenallokation nach dem gegebenen Optimierungsziel kann durch die in Tabelle 2.3 beschriebenen Scheduling-Ansätze erfolgen, für deren Umsetzung sich unterschiedliche Steuerungsarchitekturen eignen.

2.2.4 Steuerungsparadigmen und -architekturen

Aufgrund der Wichtigkeit der Produktionssteuerung für die logistische Zielerreichung in Produktionsunternehmen ist die Auswahl der Steuerungsarchitektur für die in Abschnitt 2.2.2 definierten Entscheidungskategorien einer MP und die Umsetzung einer entsprechenden Scheduling-Lösung aus den in Abschnitt 2.2.3 eingeführten Kategorien von essenzieller Bedeutung. Eine Steuerungsarchitektur für Produktionssteuerungssysteme ist nach Vogel-Heuser et al. [113, S. 154] eine Menge von Softwareelementen sowie ihre Schnittstellen und Eigenschaften, welche für die Umsetzung automatisierter Steuerungsfunktionen benötigt werden[7]. Die automatisierten Steuerungsfunktionen beziehen sich in der vorliegenden Arbeit auf die fünf zu lösenden Entscheidungskategorien. Das Gebilde aus Softwareelementen zur Implementierung dieser Steuerungsfunktionen kann auf verschiedene Arten umgesetzt werden. Dilts et al. [115] fassten 1991 vier Steuerungsarchitekturen zusammen, die sich evolutionär entwickelt haben: die zentrale, die hierarchische, die modifiziert hierarchische und die heterarchische Architektur. Diese vier Steuerungsarchitekturen blieben über die Jahre relevant, z.B. bei Leitao in [88] und auch Unland in [116, S. 28ff.]. Sie unterscheiden sich vor allem in der Zuweisungsfrage, welches Softwareelement Entscheidungen treffen darf und in der Art der Kommunikation der Softwareelemente, welche vertikal und horizontal ausgebildet sein kann [116, S. 27]. Abbildung 2.9 stellt die vier Architekturen visuell dar.

[7] Vogel-Heuser et al. orientieren sich bei ihrer Definition an der Arbeit von Clements et al. [114, S. 1].



(a) **Zentral** (b) **Hierarchisch**

(c) **Modifiziert hierarchisch** (d) **Heterarchisch**

Abbildung 2.9: Steuerungsarchitekturen, Quelle: Eigene Darstellung in Anlehnung an Unland [116, S. 28f.] und Dilts et al. [115].

Abbildung 2.9a zeigt die **zentrale Steuerungsarchitektur**. Sie besteht aus einer zentralen Entscheidungseinheit, welche die Planung und Steuerung übernimmt sowie einer Menge von passiven Softwareelementen, die Informationen vom zentralen Entscheider empfangen und ausschließlich ausführend agieren [115]. Für statische Probleme in kleinen Produktionssystemen mit kurzen und effizienten Informationswegen zwischen der Entscheidungsentität und den Passivknoten funktioniert die zentrale Steuerung gut. Hierfür bietet sie eine effektive und effiziente Problemlösung aufgrund der ganzheitlichen Optimierung aus der globalen Problemsicht [116, S. 27]. Für größere, komplexere und dynamischere Produktionssysteme ergeben sich jedoch einige Nachteile der zentralen Architektur. Durch langsame Reaktionszeiten reagiert sie träger auf Störfaktoren und ist damit weniger robust [88]. Ein Ausfall der zentralen Einheit führt sogar zum Stillstand des Gesamtsystems [116, S. 27]. Außerdem schränkt die zentrale Organisation die Skalierbarkeit des Steuerungssystems ein, da die Komplexität der größer werdenden NP-vollständigen Optimierungsprobleme exponentiell steigt [70, S. 47]. Einer zentralen Organisation entsprechen häufig Scheduling-Ansätze aus den klassischen Lösungsfeldern des dynamischen Schedulings: prädiktiv-reaktiv, robust-proaktiv und rollierend. Auch reaktive Ansätze stuft diese Arbeit als zentral ein, wenn die Prioritätsregeln zentral verwaltet werden. Für zentral berechnete Schedules finden in der Literatur hierbei häufig Genetische Algorithmen (GA) Anwendung, welche als Metaheuristik gute Näherungslösungen finden [117], [118]. GAs eignen sich demnach für kombinatorische Optimierungsprobleme wie das in Abschnitt 2.2.3 vorgestellte Scheduling-Problem [119, S. 2].

Aufgrund der genannten Defizite hat sich die **hierarchische Steuerungsarchitektur** entwickelt, welche in den 1970er Jahren den ersten Schritt in Richtung der verteilten Entscheidungsfindung darstellte. Abbildung 2.9b visualisiert die hierarchische Organisation der Entscheidungen. Höhere Level treffen strategische Entscheidungen, während niedrigere Level eher einfache lokale Entscheidungen treffen und sich an die globalen Vorgaben der obersten Entscheidungsebene halten [116, S. 27f.]. Hierarchische Beziehungen dieser Art werden auch als Master-Slave-Architektur bezeichnet, wobei sich auf niedrigeren Steuerungsebenen die Slaves an die Vorgaben des Masters halten und die entsprechend notwendigen lokalen Entscheidungen treffen [115]. Eine derartige Architektur bringt nach Unland [116, S. 28], welcher sich hierfür auf Arbeiten von Marik und Lazansky [120], Borangiu [121], Trentesaux [122] und Sallez et al. [123] bezieht, folgende Eigenschaften mit sich: Sie liefert nahezu optimale Steuerungslösungen sowie eine höhere Robustheit, Vorhersagbarkeit und Effizienz in vergleichsweise statischen Produktionsumgebungen mit niedriger Variantenvielfalt und wenigen Änderungen in der Produktion. Außerdem zeichnet sich eine hierarchische Steuerungsarchitektur durch eine geringere Störungsanfälligkeit aus.

Jedoch entwickelte sich in der praktischen Anwendung der hierarchischen Steuerung vor allem auf der Ebene der Slaves ein Bedarf an Autonomie, welcher zur **modifizierten hierarchischen Architektur** führte [115]. Um die langen vertikalen Kommunikationswege entlang der Hierarchieebenen zu umgehen, existiert zusätzlich eine horizontale Kommunikation zwischen Slaves, welche in Abbildung 2.9c dargestellt ist. Die direkte horizontale Kommunikation ermöglicht folglich schnellere Reaktionen auf Störfaktoren in der Produktion aufgrund der kürzeren Informationswege [116, S. 29f.].

Im Gegensatz zu einer hierarchisch organisierten Architektur arbeitet die **heterarchische Steuerungsarchitektur** in Abbildung 2.9d mit kollaborierenden und gleichgestellten Entscheidungseinheiten, die in keinem hierarchischen Zusammenhang stehen. Die einzelnen Entscheidungsknoten können in einer vordefinierten Weise miteinander kommunizieren und erreichen damit gemeinschaftliche Entscheidungen [116, S. 29]. Trentesaux [122] beschreibt eine Heterarchie als gerichteten Graphen aus Entscheidungsknoten, deren Master-Slave-Beziehungen durch gerichtete Kanten zwischen den Knoten beschrieben werden. Ein Spezialfall ist die vollständige Heterarchie, in der jeder Knoten die Rolle von Master und Slave einnimmt. Die vollständige Heterarchie hat hierbei Vorteile für kurzzeitige Optimierungen. In Umgebungen, die längerfristig zu optimieren sind, ist jedoch schwer nachzuweisen, dass eine vollständig verteilte Entscheidungsfindung eine ausreichende Lösungsgüte erreicht [122]. Deshalb haben sich **semi-heterarchische** Systeme bewährt, welche die Vorteile hierarchischer und vollständig heterarchischer Systeme kombinieren. Abhängig vom Anwendungsfall erfolgt hierfür die Definition der Zusammenarbeit aller Entscheidungsknoten in heterarchischer oder modifiziert hierarchischer Form [116, S. 29f.]. Für eine derartig heterarchische oder semi-heterarchische Steuerungsarchitektur gelten Multiagentensysteme (MAS) als Kerntechnologie, indem die Entscheidungsknoten durch kommunizierende Softwareagen-

ten umgesetzt werden. MAS finden sowohl für Scheduling-Probleme als auch generell in der Automatisierungstechnik erfolgreich Anwendung [88], [124]–[126]. Sie stellen eines der fünf in Abschnitt 2.2.3 eingeführten Lösungsfelder für dynamisches Scheduling dar. Abschnitt 2.3.2 führt in die Agententechnologie ein.

Zur Vereinfachung spricht diese Arbeit im weiteren Verlauf von einer **zentralen Steuerungsarchitektur** äquivalent zur hier vorgestellten zentralen Architektur und von einer **dezentralen Steuerungsarchitektur**, wenn die Architektur eine verteilte Entscheidungsfindung nach dem heterarchischen oder semi-heterarchischen Prinzip umsetzt. Die evolutionäre Entwicklung von zentralen hin zu dezentralen Architekturen zeichnet sich auch im Kontext der Industrie 4.0 ab.

2.3 Steuerung im Kontext der Industrie 4.0

Die Entwicklung vernetzter Produktionssysteme im Kontext von Industrie 4.0 rückt die Koordination und Steuerung der intelligenten Produktionsentitäten „Station“, „FTF“ und „Produkt“ in den Vordergrund und stellt hohe Anforderungen an Produktionssteuerungssysteme, die entstehenden Potenziale zu nutzen [127], [128]. Beispielsweise ermöglichen intelligente Produkte, welche in der Produktion mit Wissen über sich selbst ausgestattet und kommunikationsfähig sind, eine höhere Produktqualität und -individualität, was zu höherer Kundenzufriedenheit führt [128]. Wie die Produktionssteuerung zur Nutzung der Potenziale beitragen kann und welche Potenziale entstehen, zeigen die folgenden Abschnitte. Abschnitt 2.3.1 beschreibt die Entwicklung zu vernetzten Produktionssystemen und offenbart ihre Möglichkeiten. Abschnitt 2.3.2 legt anschließend den Fokus auf die MAS-Technologie, woraufhin Abschnitt 2.3.3 intelligente Agenten mittels Reinforcement Learning (RL) einführt. Abschließend zeigt Abschnitt 2.3.4, wie Industrie 4.0-Technologien zur Inbetriebnahme und Entwicklung der passenden Produktionssteuerung beitragen.

2.3.1 Wandel von zentralen zu dezentralen Steuerungssystemen

Nachdem die Evolution der Steuerungsparadigmen in Abschnitt 2.2.4 bereits eine Entwicklung von zentralen zu dezentralen Steuerungssystemen andeutet, zeichnet sich diese auch im Kontext der vierten industriellen Revolution ab. Xu et al. [129] heben Industrie 4.0 als notwendigen Schritt zur Digitalisierung der Produktion hervor, um als Unternehmen mit höherer Effizienz wettbewerbsfähiger zu sein. Die Digitalisierung stützt sich im Wesentlichen auf drei Technologien: das Internet der Dinge (IdD), Cloud Computing (CC) und Cyber-physische Systeme (CPS), wobei CPS als Kerntechnologie gelten.

Das IdD umfasst die Vernetzung von Dingen, z.B. Produktionsentitäten, über das Internet [130]. CC ermöglicht die Verlagerung lokaler Rechenressourcen auf entfernte Rechner, sozusagen in die Cloud [131]. CPS sind Systeme zusammenarbeitender Recheneinheiten, welche

eng mit ihrer physischen Umgebung verbunden sind [132]. Die Recheneinheiten stellen laufend Informationen über von ihnen angebotene Services im IdD zur Verfügung und greifen umgekehrt auf Informationen zu, die sie verarbeiten [133, S. 10-13], [134, S. 2f.]. Die Ursprünge von CPS lassen sich laut Park et al. [135] auf die Weiterentwicklungen im Bereich der Informations- und Kommunikationstechnologie, insbesondere der eingebetteten Systeme (engl.: embedded systems), zurückführen. Ein eingebettetes System ist ein Computersystem zur Ausführung spezifischer Funktionen, meist in Echtzeit. Hierbei gilt es als Teil eines größeren Systems oder eines Gesamtsystems. Auf Basis dieser Definition kann ein CPS auch als Netzwerk mehrerer eingebetteter Systeme bezeichnet werden, welche Daten austauschen und verarbeiten sowie physische Elemente der realen Welt überwachen und deren Sensoren und Aktoren verwalten [135].

Cyber-physische Produktionssysteme (CPPS) beschreiben die Anwendung von CPS in der Produktionsindustrie. Nach Monostori et al. [136] besteht ein CPPS aus autonomen und kooperativen Elementen sowie Subsystemen, welche über alle Level der Produktion, vom physischen Prozess über die Maschinen bis hin zu Produktions- und Logistiknetzwerken, verbunden sind. Die Elemente eines CPPS agieren autonom und intelligent, indem sie erstens Informationen über ihre Umgebung erfahren können. Zweitens bilden sie ein Netzwerk der Zusammenarbeit, indem sie ihr Wissen über Services austauschen. Drittens sind sie in der Lage, auf interne und externe Veränderungen zu reagieren [136].

Diese Fähigkeiten führen dazu, dass sich die Automatisierung in Produktionssystemen weg von der klassischen zentralen bzw. hierarchischen Automatisierungspyramide nach ISA-95 [137] hin zu einer vernetzteren, dezentraleren, CPS-basierten Automatisierung entwickelt, dargestellt in Abbildung 2.10 [136],[138].

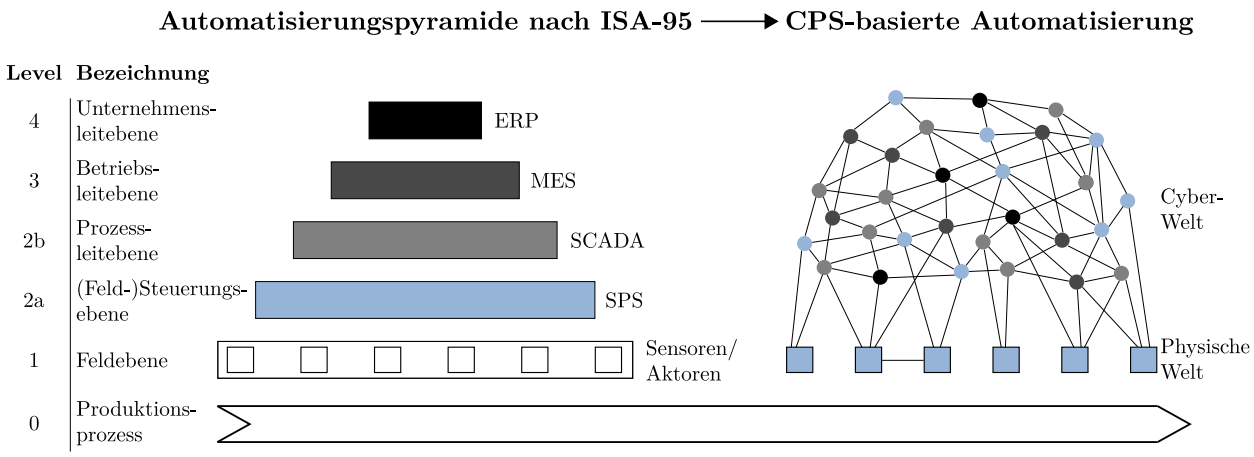


Abbildung 2.10: Wandel der Automatisierung von der Automatisierungspyramide nach ISA-95 hin zur dezentralisierten CPS-basierten Automatisierung im Kontext der Industrie 4.0, Quelle: Eigene Darstellung in Anlehnung an Monostori et al. [136] und ISA-95 [137].

Die hierarchisch organisierte Automatisierungspyramide diente für mehr als 20 Jahre als valides Modell zur Implementierung automatisierter Produktionssysteme [138]. Nach der

Norm ISA-95 [137] werden fünf Hierarchie-Level unterschieden: Level 0 Produktionsprozess, Level 1 Feldebene, Level 2 (Feld-)steuerungs- und Prozessleitebene, Level 3 Betriebsleitebene und Level 4 Unternehmensleitebene. Jede Ebene befasst sich mit unterschiedlichen Automatisierungsaufgaben und folgt bestimmten Zielen. Die vertikale Kommunikation stellt sicher, dass Entscheidungen von höheren Leveln zu niedrigeren Leveln weitergegeben werden können und Produktionsinformationen von unten nach oben ausgetauscht werden. Je höher die Ebene, desto abstrakter und strategischer ist der Blick auf die Produktion [138], [139, S. 36ff.].

Von oben beginnend verwaltet die Unternehmensleitebene (Level 4) langfristig strategische Aufgaben zur Unternehmensführung. Hierzu zählen die Grobplanung der Produktion hinsichtlich Materialverbrauch, Anlieferung, Versand, Lagerbeständen und Betriebsmanagement, durchgeführt mithilfe von Enterprise Resource Planning (ERP)-Systemen. Die Betriebsleitebene (Level 3) plant mittelfristig die Arbeitsabläufe und Produktionspläne der gewünschten Endprodukte und ist für die Datenpflege und Optimierung der Produktionsprozesse sowie die Feinplanung und Betriebssicherheit verantwortlich. Geplant wird mit einem Manufacturing Execution System (MES) [101], [137]. Auf dieser Ebene findet damit der Scheduling-Prozess in enger Abstimmung mit den logistischen Zielgrößen, vorgegeben von der Unternehmensleitebene, statt [140]. Die Prozessleit- und (Feld-)Steuerungsebene (Level 2a und b) überwachen und koordinieren die kurzfristige Steuerung der physischen Produktionsprozesse. Hierzu werden sogenannte „Supervisory Control and Data Acquisition“ (SCADA)-Systeme sowie SPS eingesetzt, die u.a. Rückmeldung über scheduling-relevante Informationen an das MES-System geben. Die SPS sind direkt mit den Sensoren und Aktoren der Feldebene (Level 1) verbunden, auf welcher die physischen Prozesse gemessen und eingestellt werden. Dies geschieht im Zeitrahmen von weniger als einer Sekunde. Level 0 repräsentiert die tatsächlich stattfindenden physischen Produktionsprozesse [101], [137].

Ein CPPS bricht diese Automatisierungspyramide teilweise auf. Die Level 0, 1 und 2a aus Abbildung 2.10 bleiben erhalten. SPS steuern die Aktoren und lesen die Sensoren, um für die zeitkritischen physischen Produktionsprozesse die höchstmögliche Performanz in Echtzeit zu erzielen. Monostori et al. [136] sprechen hierbei von der physischen Welt. Auf höheren und strategischeren Ebenen ist jedoch ein dezentralisiertes, nicht länger hierarchisches System charakteristisch. Intelligentes Datenmanagement, Datenanalyse und die darauf basierenden Steuerungsentscheidungen finden dezentral in der Cyber-Welt statt [136].

Die Entwicklung von der zentralen pyramidenförmigen Welt hin zur dezentralisierten CPPS-Welt hat Einfluss auf die zu treffenden Scheduling-Entscheidungen, weil sowohl die Entscheidungen als auch die Soft- und Hardware-Entitäten, die die Entscheidungen herbeiführen, dezentralisiert werden. Dies erfordert den Austausch von Informationen in Echtzeit sowie neue Scheduling-Strategien, die die CPPS-Welt in die Lage versetzen, die vielversprechenden schnellen Reaktionszeiten und die hohe Robustheit gegenüber Störfaktoren in der Produktion zu nutzen [141]. In Kombination mit dem IdD und CC ermöglichen CPS die

Sammlung der notwendigen Echtzeitinformationen aus der Produktion. Davon profitieren wiederum die dynamischen Scheduling-Algorithmen, welche die aktuellen Produktionsdaten zu Steuerungsentscheidungen verarbeiten [101]. Das Gesamtkonstrukt eines intelligenten Produktionssystems wird dann als „Smart Factory“ zusammengefasst [127], [142]. Weiterführende Informationen zu Smart Factory und Industrie 4.0 geben beispielsweise [143]–[145]. Die Umsetzung der modernen Scheduling-Strategien für CPPS erfolgt i.d.R. mittels Softwareagenten. Darauf geht der nächste Abschnitt ein.

2.3.2 Multiagentensysteme

Zur Umsetzung dezentraler Steuerungsarchitekturen in CPPS beschreiben Vogel-Heuser et al. [146] Agenten als wesentlichen Enabler, um die vernetzten Produktionsentitäten und deren Kommunikation zu modellieren. Für den Begriff „Agent“ existiert keine einheitliche Definition. Nach Wooldridge [147] kann ein Agent als eine autonome, problemlösende und zielorientierte Recheneinheit mit sozialen Fähigkeiten betrachtet werden. Diese Einheit ist in der Lage, sich in einem offenen und dynamischen Umfeld effektiv, teilweise sogar proaktiv zu verhalten. Der Agent beobachtet hierzu die Umgebung und handelt anhand seiner Beobachtungen, um seine Ziele zu erreichen [147, S. 21f.], [148]. Im Einklang mit Wooldridges Definition, aber weniger spezifisch, ist die Definition von Dorri et al. [149]. Sie definieren einen Agenten als eine Entität, die in einer Umgebung agiert, welche der Agent durch Parameter wahrnimmt. Auf Basis dieser Umgebungsparameter entscheidet sich der Agent für im Einklang mit seinen Zielen stehende Aktionen. Nach Unland [150, S. 4] zeichnet sich ein derartig agierender Agent durch die Eigenschaften Autonomie, Reaktionsfähigkeit, Proaktivität, kluges Verhalten, soziale Fähigkeit und Lernfähigkeit in seiner Umgebung aus. Dorri et al. [149] heben hiervon die soziale Fähigkeit, die Autonomie und die Proaktivität von Agenten hervor und begründen damit ihre breite Anwendung und ihre Fähigkeit, komplexe Aufgaben zu lösen. Tabelle 2.4 fasst die Eigenschaften eines Agenten zusammen, mit denen ein Agent je nach Konfiguration ausgestattet sein kann.

Aufgrund der begrenzten Fähigkeiten eines einzelnen Agenten erfordern komplexere Probleme in der realen Welt die gemeinsame und kooperative Anstrengung mehrerer Agenten zur Lösung des Problems. Erst in einem MAS kommen deshalb die wesentlichen Vorteile der Agententechnologie zum Tragen [149]. Dabei ist ein MAS ein Zusammenschluss von voll- oder halbautonomen Agenten, die kollaborieren, um ihre individuellen Ziele in Einklang zu bringen sowie die Gesamtziele des Agentenverbunds zu erreichen. Basierend auf der sozialen Fähigkeit der einzelnen Agenten ist ein erfolgreiches MAS auf Kommunikation, Zusammenarbeit, Verhandlung sowie auf Delegation von Verantwortung angewiesen [150, S. 6], [152]. Sind diese Aspekte erfüllt, zeichnet sich ein MAS nach Unland [150, S. 7] durch die Eigenschaften Dezentralisierung, Flexibilität, Anpassungsfähigkeit, Skalierbarkeit, Schlankheit und Robustheit aus, welche Tabelle 2.5 beschreibt.

Tabelle 2.4: Eigenschaften eines Agenten nach [150, S. 4], [149], [148], [151].

Eigenschaft	**Beschreibung**
Autonomie	Anhand seines Verhaltens hat der Agent die alleinige Kontrolle über seinen inneren Zustand und seine Ziele.
Reaktionsfähigkeit	Sensoren und Aktoren bilden die Schnittstelle des Agenten zu seiner Umgebung. Auf sensorische Inputs reagiert er mit seinen Aktoren entsprechend seiner Ziele.
Proaktivität	Ein Agent kann proaktiv mögliche Veränderungen in seiner Umgebung antizipieren, darauf reagieren sowie Aktionen entsprechend seiner Ziele einleiten.
Ergebnisorientierung	Ein Agent agiert seinen Zielen entsprechend und ergreift die Initiative, wenn er dafür eine Möglichkeit sieht.
Kluges Verhalten	Durch umfassende Fachkenntnisse in einem genau definierten Bereich ist ein Agent in der Lage, Probleme in diesem Bereich zu lösen.
Soziale Fähigkeit	Zur Erreichung seiner Ziele interagiert ein Agent direkt mit Menschen oder auch anderen Agenten.
Lernfähigkeit	Damit ein Agent anpassungsfähig sein kann, muss er in der Lage sein, aus seinen Erfahrungen zu lernen.

Die „Foundation for Intelligent Physical Agents“ (FIPA) hat sich zur Aufgabe gemacht, die Zusammenarbeit industrieller MAS zu spezifizieren, um vor allem die Interoperabilität im Agentenbereich zu begünstigen und Entwicklungsartefakte zu vermeiden. Zum FIPA-Modell zählen Standards zu Themen wie Semantik und Syntax von Nachrichten zwischen Agenten, Agentenmanagement und Nachrichtentransport. Wesentlich für die Zusammenarbeit in MAS sind Protokolle, die ihre Kollaboration spezifizieren [153]. Für das Produktions-Scheduling wird hiervon häufig das sogenannte „Contract Net Protocol“ (CNP) verwendet [154]–[159], dessen Anwendbarkeit für die dezentrale, adaptive Produktionssteuerung mit dem großen Vorteil des robusten Umgangs mit Produktionsdynamiken schon 1987 bestätigt wurde [160]. Entwickelt und vorgestellt wurde das verhandlungsbasierte CNP bereits im Jahr 1980 von Smith [161], welcher die Art der Verhandlung nach dem CNP auch als Metapher für dezentrale Problemlösung bezeichnet [162]. In dieser Arbeit ist das CNP somit ein potenzieller dezentraler Lösungsmechanismus des betrachteten Scheduling-Problems und beschreibt gleichzeitig, wie Agenten mittels Auktionen zu vertraglicher Übereinkunft gelangen können. Abschnitt 4.2 geht im späteren Verlauf der Arbeit näher auf das CNP ein.

Für einfacher gestaltete Interaktionen zwischen Agenten können auch sogenannte Request-Response-Sequenzen verwendet werden. Hierbei stellt ein Agent eine Informationsanfrage an einen anderen Agenten, welcher mit der gewünschten Information antwortet [147, S. 157].

Neben reinen MAS haben sich auch sogenannte holonische Steuerungssysteme entwickelt. Sie setzen auf die MAS-Technologie und wenden sie zur Umsetzung kollaborierender Holone

Tabelle 2.5: Eigenschaften eines MAS nach [150, S. 7].

Eigenschaft	**Beschreibung**
Dezentralisierung	Aufgrund der Autonomie seiner Agenten verfügt ein MAS immer über eine dezentralisierte Struktur und Kontrolle.
Flexibilität	Ein MAS reagiert flexibel auf unvorhergesehene plötzliche Störfaktoren wie z.B. Maschinenstörungen. Bei Nichtverfügbarkeit kann ein Agent durch andere ersetzt oder kompensiert werden.
Anpassungsfähigkeit	Ein MAS kann sich an sich ändernde Gegebenheiten und Aufgaben anpassen. Es rekonfiguriert sich, indem neue und geeignetere Agenten eintreten, andere gegebenenfalls austreten, wodurch sich die Qualität und Funktionalität des MAS verbessert.
Skalierbarkeit	Ein MAS ist ein verteiltes Computersystem. Die Computer sind über ein Netzwerk miteinander verbunden. Das Hinzufügen neuer Agenten in Form von Computern ist eine implizit vorhandene Eigenschaft eines MAS.
Schlankheit	Um die Komplexität einzuschränken und das Verhalten eines MAS verstehen zu können, decken Agenten eine klar definierte und begrenzte Problemdomäne ab.
Robustheit	Autonomie und Intelligenz in einem MAS fördern ein hohes Maß an Robustheit und Fehlertoleranz gegenüber Hard- und Softwareproblemen, zum Beispiel bei einer Stationsstörung in der MP.

an. Dabei adressieren holonische Systeme eine Mischform zentraler und dezentraler Entscheidungsfindung, wobei der Fokus auf der Dezentralität durch verteilte Entscheidungsfindung liegt [88]. Deshalb zählen sie in dieser Arbeit nach der Nomenklatur aus Abschnitt 2.2.4 zu den dezentralen Ansätzen. Der Begriff Holon stammt von Köstler [163]. Er nutzte sie zur Beschreibung der Organisation in lebenden Organismen und sozialen Organisationen, wobei ein Holon sich einerseits durch Autonomie und andererseits durch Kooperation und damit als Teil eines größeren Ganzen charakterisiert. In Produktionssystemen repräsentiert ein Holon eine Entscheidungsentität der Produktion, beispielsweise eine Maschine oder einen Produktionsauftrag. Ein System mehrerer Holone wird als Holarchie bezeichnet. Eine Holarchie kooperiert in einer hierarchischen Art und Weise zur Erreichung von Produktionszielen, wobei ein Holon gleichzeitig mehreren Holarchien zugeordnet sein kann. Durch die Autonomie einzelner Holone sowie die Organisation hierarchischer Holarchien vereinen holonische Steuerungssysteme die in Abschnitt 2.2.4 beschriebenen Vorteile hierarchischer und heterarchischer Ansätze [88].

Über die Jahre wurden einige holonische Steuerungsarchitekturen entwickelt. Die wohl bekanntesten sind die „Product-Resource-Order-Staff Architecture“ (PROSA) [164] und die „ADAptive holonic COntrol aRchitecture“ (ADACOR) [165]. Beide werden in Abschnitt 4.1

beschrieben. Bis heute dienen sie als Basis für einige problemspezifische Weiterentwicklungen holonischer Architekturen. Beispielsweise entwickelten Mönch et al. [166] eine hierarchisch organisierte holonische Steuerung für komplexe Halbleiterfertigungssysteme mit dem Namen FABMAS, welche auf der PROSA-Architektur basiert. Das holonische MAS steuert ein in AutoSched AP implementiertes Simulationsmodell auf den Ebenen „Fabrik“, „Arbeitsbereich“ und „Arbeitszentrum“. Eine der vielversprechendsten Weiterentwicklungen von PROSA oder ADACOR ist die dynamische Anpassung des Holonenverhaltens an Veränderungen des Produktionssystems, wodurch Produktionseffizienzen gesteigert werden können [167], [168]: beispielsweise in [169] als Weiterentwicklung zu ADACOR und in [170] auf Basis beider Architekturen. Einen Überblick zu weiteren holonischen Steuerungssystemen geben u.a. Leitao [88] und Derigent et al. [168].

Dezentrale MAS- sowie holonische Steuerungen bringen zusammenfassend einige vorteilhafte Eigenschaften wie Robustheit und Autonomie für z.B. einen effektiven Umgang mit Stationsstörungen bei der Steuerung der MP mit. Als weiterer vielversprechender Aspekt gilt die Lernfähigkeit von Agenten.

2.3.3 Lernende Agenten mittels Reinforcement Learning

Eine der Eigenschaften von Agenten aus Tabelle 2.4 ist die Lernfähigkeit. Sie stellt einen der Kernaspekte für intelligentes Agentenverhalten dar [171, S. 199]. Die Lernfähigkeit wird vor allem in sehr komplexen Umgebungen relevant, in der fest programmiertes Agentenverhalten schwer umzusetzen ist, weil beispielsweise nicht alle Informationen zur Umgebung des Agenten zur Verfügung stehen [172, S. 1]. Lernfähige Agenten können entweder als einzelne Agenten agieren oder als lernendes MAS auftreten [171, S. 199]. Nach Abschnitt 2.2.4 agiert ein einzelner Agent zentral und ein lernendes MAS dezentral.

Wooldridge [147, S. 34] beschreibt ein Modell für intelligent agierende Agenten. Er formalisiert einen Agenten mit einer finiten Menge ihm zur Verfügung stehender Aktionen, anhand derer der Agent mit seiner Umgebung sequenziell interagiert. Durch jede Aktion des Agenten ändert sich der Status seiner Umgebung. Sowohl aus spieltheoretischer Sicht [171, S. 215-220] als auch aus Sicht der Künstlichen Intelligenz (KI) [173, S. 25, S. 103] eignet sich für derartig formalisierte sequenzielle Entscheidungsprobleme RL als Methode. Vor allem in komplexen Entscheidungsproblemen, wie der des Produktions-Schedulings bzw. der Steuerung der MP, ist RL mit den Rechenleistungen heutiger Computer eine vielversprechende Methode, intelligente Agenten zu implementieren und die Ressourcenallokation im Produktionssystem Aktion für Aktion zu lernen und anschließend durchzuführen [174], [175]. Dies bestätigen auch Ergebnisse aus der Spieleindustrie, wo komplexe sequenzielle Entscheidungsfolgen in Spielen wie „SpaceInvaders“, „Schach“ oder „Go“ zu lösen sind. Hier erzielten ein oder mehrere RL-Agenten zahlreiche Erfolge [176], [177].

In Einklang mit Wooldridges [147, S. 34] Modell für intelligente Agenten entspricht RL

einem sequenziellen Lernvorgang anhand des „trial-and-error“-Prinzips[8], dargestellt in Abbildung 2.11 [178, S. 47f.]. Das lernende Computerprogramm wird als Agent bezeichnet. Dieser Agent ist der Entscheidungsträger, der mit seiner Umgebung interagiert und in dieser Entscheidungen trifft. Die Umgebung umfasst alles außerhalb des Agenten und damit alles, worüber der Agent keine absolute Kontrolle hat. Die Umgebung wird durch eine Reihe von Variablen repräsentiert, die mit dem zu lösenden Problem zusammenhängen. Diese Menge an Variablen und alle möglichen Werte, die sie annehmen können, werden als Zustandsraum bezeichnet. Ein Zustand (engl.: state) s_t in einem Zeitschritt t ist eine Instanz des Zustandsraums, ein Satz von Werten, die die Variablen zum aktuellen Zeitpunkt annehmen[9]. In jedem State[10] verfügt der Agent über eine Menge an Aktionen (engl.: actions) a_t, aus denen er wählen kann. Durch diese Actions[11] beeinflusst der Agent die Umgebung. Als Reaktion auf die Action des Agenten ändert sich der State des Systems zu s_{t+1}. Die Umgebung liefert auch ein Belohnungssignal (engl.: reward) r_{t+1} an den Agenten, welches die Güte der Action und damit der Entscheidung des Agenten repräsentiert. Die Wechselwirkungen zwischen dem Agenten und der Umgebung dauern mehrere Zyklen lang an. Jeder Zyklus umfasst einen Zeitschritt t. Bei jedem Zeitschritt beobachtet der Agent wie beschrieben die Umgebung, ergreift eine Action und erhält einen Reward[12]. So sammelt der Agent Erfahrungen, welche ihm die Möglichkeit bieten, zu lernen und die Leistung zu verbessern [179, S. 6-10], [178, S. 47ff.]. Speichert der Agent das gesammelte Wissen in einem tiefen Neuronalen Netz (NN), spricht die Literatur von Deep Reinforcement Learning (DRL) [178, S. 475].

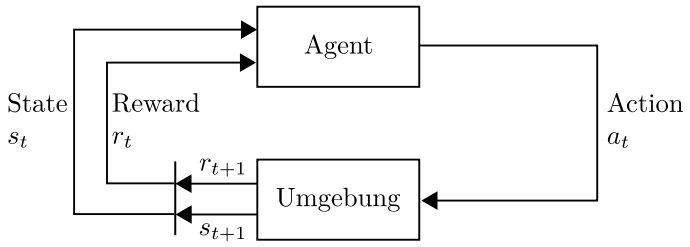


Abbildung 2.11: Agent lernt durch Interaktion mit seiner Umgebung, Quelle: Eigene Darstellung in Anlehnung an Sutton und Barto [178, S. 48].

Neben den bereits in Abschnitt 2.2.4 eingeführten **zentralen** und **dezentralen Steuerungsansätzen** fasst diese Arbeit RL-Ansätze unter dem Begriff der **lernenden Steuerung** zusammen.

[8]Für eine prägnante Darstellung der RL-Grundlagen auf Basis der vorwiegend englischsprachigen Literatur verwendet die vorliegende Arbeit im RL-Kontext im Verlauf der Arbeit einige englische Fachbegriffe.
[9]Obwohl s_t häufig ein Vektor mehrerer Werte ist, verwendet die Literatur hier keine Vektorschreibweise.
[10]State wird im Zusammenhang mit RL zur Integration in den deutschen Text ab hier großgeschrieben.
[11]Action wird im Zusammenhang mit RL zur Integration in den deutschen Text ab hier großgeschrieben.
[12]Reward wird im Zusammenhang mit RL zur Integration in den deutschen Text ab hier großgeschrieben.

2.3.4 Virtuelle Inbetriebnahme von Steuerungssystemen

Bevor Steuerungsalgorithmen, ganz gleich ob zentral, dezentral oder lernend, in reale Produktionssysteme überführt werden können, ist deren Funktionalitätsprüfung und Validierung erforderlich. Auch hier existiert ein Trend, dem im Kontext von Industrie 4.0 mehr Bedeutung zukommt: die Virtuelle Inbetriebnahme (VIBN) [180]–[182].

Nach Reinhart und Wünsch [183] ist die VIBN ein Ansatz zur Inbetriebnahme von Steuerungssoftware durch virtuelle Prototypen. Die VIBN wird bereits in einer frühen Phase und damit parallel zur Entwicklung des eigentlichen Produktionssystems eingesetzt. Dadurch kann die Software in einem frühen Entwicklungsstadium auf Funktionalität geprüft und validiert werden. Kosten für späte Modifikationen eines ausgereiften Steuerungssystems werden somit vermieden. Die Hochlaufzeiten der Produktion verkürzen sich [184, S. 2-7], [183].

Abhängig von der Umsetzung des Steuerungssystems in der VIBN lassen sich im Allgemeinen zwei Arten von VIBN unterscheiden: simulativ und real. Wird die Steuerungslogik virtuell abgebildet, sprechen Reinhart und Wünsch [183] von rein simulativer VIBN. Je nach Modellierungssprache für die Steuerungslogik differenziert die VDI/VDE-Norm 3693 [184, S. 8f.] weiter in „Model-in-the-Loop“ (MIL) für prototypisch umgesetzten Steuerungscode und „Software-in-the-Loop“ (SIL) für virtualisierten Seriencode. „Hardware-in-the-Loop“ (HIL) bezeichnet hingegen die zweite Art der VIBN: die Inbetriebnahme realer Steuerungssoft- und Hardware. Die reale Soft- und Hardware wird hierfür an eine virtuelle Produktionsumgebung angebunden und dort getestet [183].

Der häufigste Anwendungsfall der VIBN ist die Inbetriebnahme einer SPS, beispielsweise in [185], wobei eine reale SPS nach dem HIL-Prinzip getestet wird. Entlang der klassischen Automatisierungspyramide aus Abbildung 2.10 findet die VIBN auf weiteren Ebenen Anwendung. Die Arbeit von Seidel et al. [186] befasst sich z.B. mit VIBN auf Prozessleitebene. Die Autoren nehmen ein Materialflusssystem in Betrieb, welches mithilfe einer ereignisorientierten Simulation (engl.: discrete-event simulation - DES) simuliert wird. Schleipen et al. [187] befassen sich hingegen mit VIBN von MES-Systemen auf Betriebsleitebene. Sie testen die Steuerungssoftware anhand einer Digitalen Fabrik, um spät verursachte Kosten durch Steuerungsfehler zu vermeiden.

Für eine erfolgreiche VIBN beschreiben Zäh et al. [188] die Notwendigkeit der Kopplung eines Verhaltensmodells mit der Steuerungslogik über eine Schnittstelle am Beispiel einer Montagestation. Daraus lässt sich für die MP ein dreiteiliges Vorgehen ableiten: Zunächst ist das Produktionssystem anhand einer **Simulation** virtuell abzubilden, um die Abläufe der geplanten Produktion simulieren zu können. Weiterhin ist die **Steuerungslogik** so zu modellieren, dass sie der geplanten physischen Darstellung entspricht. Zuletzt erfordert die VIBN eine **Schnittstelle** für Datenaustausch zwischen Simulation und Steuerung.

Dieses dreiteilige Vorgehen wählen Mönch et al. in [189] für eine VIBN von Steuerungssoftware in einer Halbleiterfertigung. In ihrer Implementierung bildet ein Simulationsmodell

der Halbleiterfertigung die Produktionsabläufe ab. Die Daten der Produktion werden in einem Datenmodell gespeichert, auf welches die Steuerungssoftware Zugriff hat. Über Trigger-Events wie Maschinenausfälle kann die Berechnung neuer Schedules angestoßen werden. Außerdem ermöglicht das Framework zeitlich periodische Trigger für das rollierende Scheduling, um regelmäßig neue Schedules berechnen zu können. Zur Berechnung der Schedules nutzt die Steuerungssoftware die im Datenmodell hinterlegten Produktionsdaten. Das Datenmodell entkoppelt Simulation und Steuerung und bildet damit die Schnittstelle zwischen Simulation und Steuerungslogik.

Weitere Beispiele der simulationsbasierten Entwicklung und Evaluation von Steuerungsalgorithmen finden sich in Negahbans und Smiths [190] Arbeit mit Fokus auf die DES als Simulationsmethode. Demnach findet die Simulation vermehrt Anwendung bei der Entwicklung von Steuerungssystemen. Hierbei treten auch Herausforderungen auf. Fowler und Rose [191] beschreiben beispielsweise die Interoperabilität mehrerer Simulationsmodelle sowie deren Interoperabilität mit Zusatzsoftware wie MES-Systemen als große Herausforderung. Schreiber [192] adressiert hingegen die Herausforderung der Vergleichbarkeit unterschiedlicher Steuerungsarchitekturen unter gleichen Rahmenbedingungen. In seiner Dissertation erarbeitet er ein Vergleichsframework für dezentrale und zentrale Steuerungsarchitekturen. Davon erhofft sich Schreiber, den Durchbruch dezentraler Architekturen zu fördern.

Ebenfalls die Vergleichbarkeit zentraler und dezentraler Steuerungsalgorithmen streben die Autoren in [37] nach dem MIL-Prinzip für die Steuerung der MP an. Ein konfigurierbares Simulationsmodell in der DES-Software Tecnomatix Plant Simulation simuliert hierbei die Produktionsabläufe der MP. Stationen können in einer Rasterstruktur angeordnet und auf beliebige Stationszahlen skaliert werden. Dies ermöglicht die Simulation verschieden konfigurierter Produktionssysteme. Die Fahrzeuge bewegen sich auf Zweibahnstraßen. Über eine „Transmission Control Protocol/Internet Protocol“ (TCP/IP)-Schnittstelle tauscht das Simulationsmodell Informationen mit der Steuerungslogik aus, ähnlich zur Umsetzung von Sundermeier et al. in [193]. Anhang A.1 dokumentiert weitere Details der Implementierung.

Derartige VIBN-Frameworks dienen als Grundlage für die spätere Implementierung und Evaluation der Steuerungssysteme, um realistische Handlungsempfehlungen zu ermöglichen. Zur Entwicklung der Steuerungsalgorithmen führt der folgende Abschnitt das gesamte Grundlagenkapitel zusammen und leitet eine konkrete Problembeschreibung ab.

2.4 Zusammenfassung der Steuerung der Modularen Produktion

Abschnitt 2.1 zeigt, dass der bisherige Fokus der MP-Forschung in der Automobilindustrie auf ihrer Planung und Ausgestaltung liegt. Gleichzeitig stellt sich die Wichtigkeit der Steuerung einer MP heraus, um ihre Potenziale an Flexibilität und höherer Produktionseffi-

zienz zur Optimierung logistischer Zielgrößen ausschöpfen zu können. Deshalb definiert Abschnitt 2.2 anschließend das Steuerungsproblem der MP. Die fünf zu lösenden Entscheidungskategorien Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement werden hierzu als Scheduling-Problem formuliert. Zur Lösung des formalen Scheduling-Problems beschreibt der anschließende Abschnitt die Evolution von zentral organisierten hin zu dezentralen Steuerungsparadigmen, die in der Literatur Anwendung finden. Im Kontext der Industrie 4.0 in Abschnitt 2.3 verdeutlicht sich diese Entwicklung zu einer vernetzteren dezentral organisierten und auch intelligent lernenden Produktionssteuerung, weg von starr zentral organisierten Systemen.

Auf dieser Basis gibt Abschnitt 2.4.1 einen das Grundlagenkapitel abschließenden Überblick zu zentralen, dezentralen und lernenden Algorithmuskonzepten und bezieht sich auf die Forschungsfragen der Dissertation (vgl. Abschnitt 1.2). Abschließend schlussfolgert Abschnitt 2.4.2 die zur Beantwortung der Fragen betrachtete konkrete Problemstellung.

2.4.1 Zentrale, dezentrale und lernende Steuerungssysteme

Das Steuerungsproblem der MP umfasst die Auftragseinlastung, das Stationsmanagement, das Auftragsrouting, das Flottenmanagement und das Störungsmanagement. Die Abschnitte 2.2 und 2.3 leiten die zur Steuerung der MP verfügbaren Steuerungsarchitekturen ab. Damit adressieren sie die **erste Forschungsfrage** mit dem Thema, welche Steuerungsarchitekturen für die MP grundsätzlich zur Verfügung stehen. Die geeigneten Paradigmen zeichnen sich dabei durch unterschiedliche Eigenschaften aus. Zentrale Ansätze optimieren gesamtheitlich und versprechen damit eine höhere Lösungsgüte der Entscheidungen. Dezentralen Steuerungssystemen basierend auf der MAS-Technologie wird häufig eine hohe Robustheit gegenüber Störfaktoren in der Produktion zugeschrieben. Lernende Ansätze zeichnen sich hingegen durch ihre Anwendung in komplexen Umgebungen aus, in denen nicht ausreichend viele Informationen über die Produktion zur Verfügung stehen, um z.B. dezentrales Agentenverhalten fest zu programmieren.

Zur Analyse der Eignung der Steuerungsparadigmen für die MP sowie deren Umsetzung entwickelt diese Arbeit, der Steuerungsevolution folgend, einen zentralen (vgl. Kapitel 3), einen dezentralen (vgl. Kapitel 4) und einen lernenden (vgl. Kapitel 5) Steuerungsansatz, zusammengefasst in Abbildung 2.12. Die auf dem jeweiligen Stand der Forschung basierende Konzeptionierung der zentralen, dezentralen und lernenden Steuerung adressiert damit im Laufe der Arbeit die **zweite Forschungsfrage**.

Abschließend stellt sich die **dritte Forschungsfrage**, welcher der entwickelten Ansätze sich in Abhängigkeit der Produktionskonfiguration sowie der Beschaffenheit von Ressourcenstörungen für die MP am besten eignet. Kapitel 6 befasst sich deshalb mit dem Vergleich der drei Steuerungen und untersucht die Algorithmuskonzepte nach ihrer Eignung für die MP in der abschließenden Evaluation.

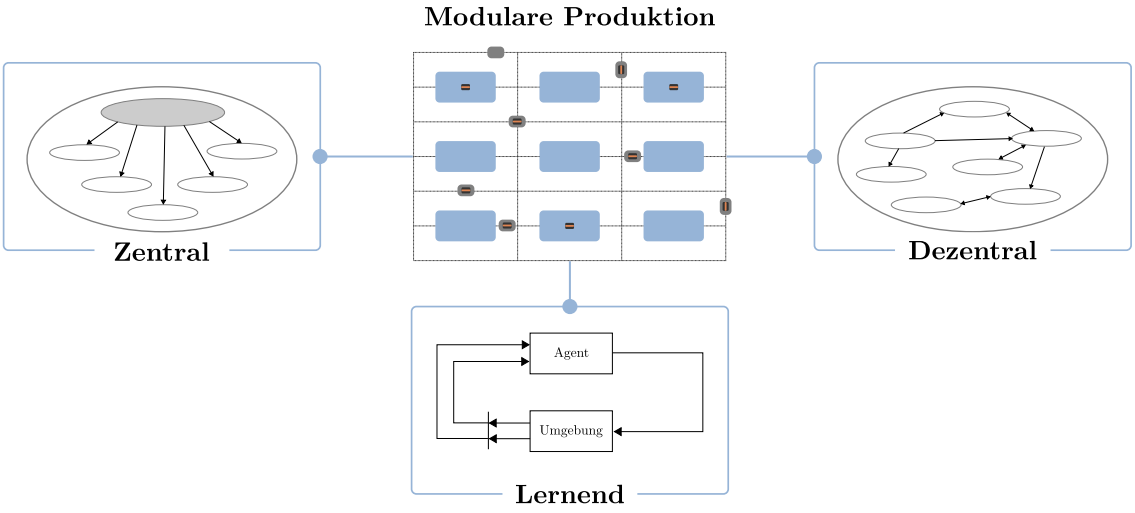


Abbildung 2.12: Entwicklung einer zentralen, dezentralen und lernenden Steuerung für die MP.

2.4.2 Problembeschreibung, Anforderungen und Annahmen

Die Beantwortung der zweiten und dritten Forschungsfrage erfordert eine konkrete Problembeschreibung. Bisher wurde eine einzelne Station der MP als Blackbox betrachtet. Je nach Anwendung und abhängig von der Art der gefertigten Produkte unterscheiden sich modulare Produktionssysteme vor allem durch ihren Stationsaufbau. Die vorliegende Arbeit orientiert sich hierbei an einer von der Audi AG bereits in Betrieb genommenen MP für die Montage von E-Achsantrieben. Die produzierten E-Achsantriebe werden hierbei auf WTs von Station zu Station transportiert und bearbeitet. Abgeleitet von der Realität stellt Abbildung 2.13 den praxisnahen Aufbau einer modularen Station dar.

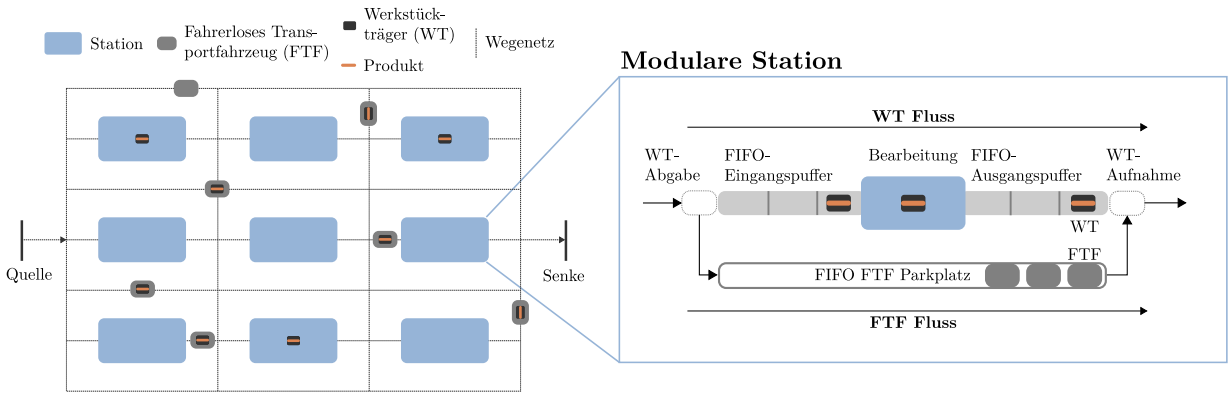


Abbildung 2.13: MP in einer rasterförmigen Anordnung von Bearbeitungsstationen umgeben von einem Wegenetz für den Materialtransport mittels FTFs, wobei die Stationen aus einem EP für Produkte tragende WTs, einem Bearbeitungsplatz, einem AP für die fertig bearbeiteten Produkte sowie einem Parkplatz für Fahrzeuge bestehen.

Am Eingang der Station gibt ein Fahrzeug das Produkt inklusive WT an das als Fließband gestaltete Eingangspufferband mit Kapazität n_{EP} ab. Der Puffer folgt dem First-In-First-

Out(FIFO)-Prinzip. Das FTF hat die Möglichkeit, in einer dafür vorgesehenen Parkzone zu parken. Ist der Bearbeitungsplatz frei, kann ein im EP wartender WT mithilfe des Fließbands auf die Bearbeitungsposition fahren, um dort bearbeitet zu werden. Nach der Bearbeitung fördert ein weiteres Fließband den WT in den AP mit Kapazität n_{AP} aus, der wiederum nach dem FIFO-Prinzip arbeitet. Es folgt die Aufnahme des WTs durch ein FTF und der Weitertransport zur nächsten Station. Je größer die Kapazität des EPs oder APs, desto länger ist die Transportstrecke zwischen WT-Abgabe und dem Bearbeitungsplatz bzw. zwischen dem Bearbeitungsplatz und der WT-Aufnahme. Deswegen korreliert die minimale Transportzeit (Puffer frei) auf dem jeweiligen Pufferband mit der Pufferkapazität und hängt von der fließbandbedingten Transportzeit pro Pufferplatz t_{PP} ab. Für die Mindest-Transportzeiten in EP und AP gilt damit: $t_{\mathrm{EP},min} = n_{\mathrm{EP}} \cdot t_{\mathrm{PP}}$ und $t_{\mathrm{AP},min} = n_{\mathrm{AP}} \cdot t_{\mathrm{PP}}$.

Mit diesem Hintergrund fasst Abbildung 2.14 den Betrachtungsumfang dieser Dissertation zusammen. Die in der Arbeit entwickelten Steuerungsalgorithmen werden darauf ausgelegt, das in Abschnitt 2.2.3 eingeführte Scheduling-Problem unter Berücksichtigung der in Abschnitt 2.2.2 formulierten Entscheidungskategorien für die in Abschnitt 2.1.2 eingeführte Art der MP zu lösen. Optimierungsziel ist ein Maximum im mittleren Durchsatz D_m.

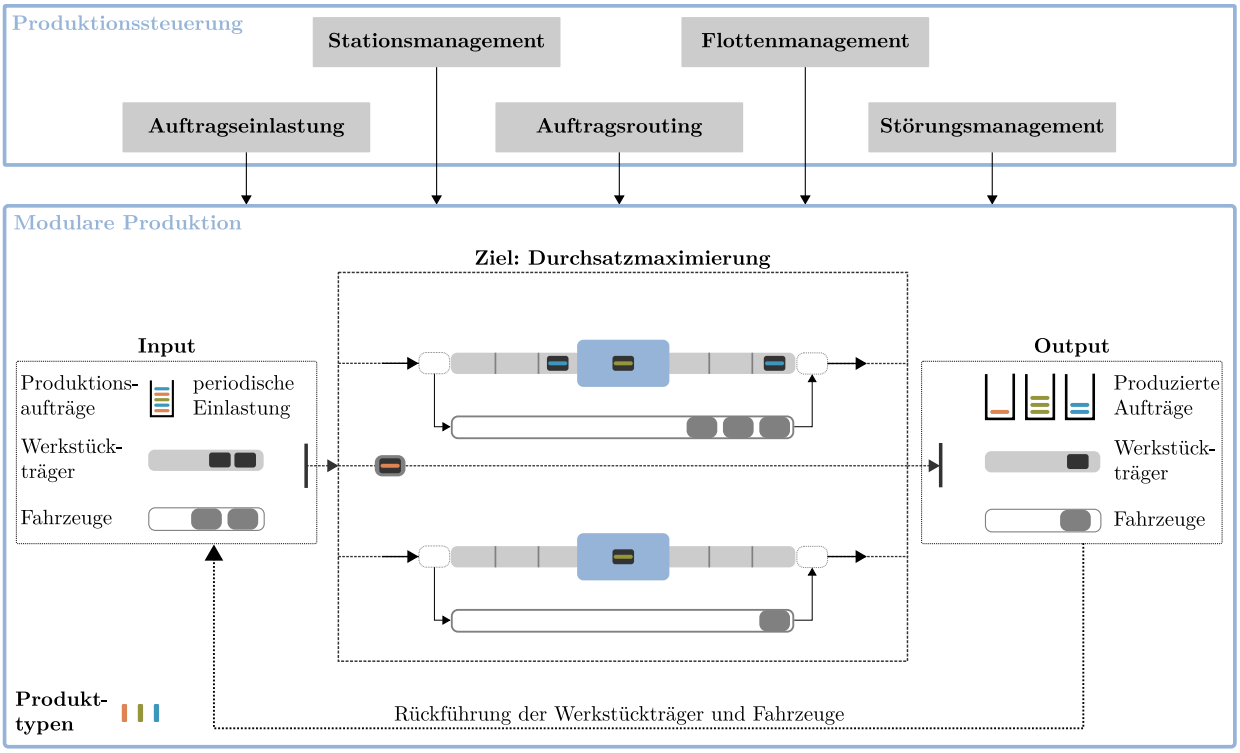


Abbildung 2.14: Visuelle Zusammenfassung der Problembeschreibung.

Die folgenden Annahmen konkretisieren den Betrachtungsumfang in den fünf Entscheidungskategorien. Sie orientieren sich an den vorgestellten Konzepten in Abschnitt 2.1.1 und schränken den Umfang so weit ein, dass die Aussagekraft der späteren Algorithmus-

untersuchungen für eine breite Masse an MP-Konzepten gegeben ist, der Steuerungsumfang gleichzeitig aber handhabbar wird: Die **Auftragseinlastung** geht von einem vordefinierten Produktmix aus. Beispielsweise sollen jeweils 100 Produkte von drei verschiedenen Produkttypen an einem Tag produziert werden, wobei sie in einem 1:1:1-Verhältnis einzulasten sind. Die Produktionsplanung stellt eine dem Produktmix entsprechende periodische Einlastungsreihenfolge in Form eines Produktionsprogramms als Empfehlung bereit. Bei drei Produkttypen A, B und C sieht diese beispielsweise folgendermaßen aus: $B \rightarrow C \rightarrow A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow \ldots$ etc. Dabei ist von der Steuerung zumindest das Einlastungsverhältnis (hier 1:1:1) einzuhalten. Auch $A \rightarrow C \rightarrow B \rightarrow C \rightarrow B \rightarrow A \ldots$ wäre legitim, wenn die Steuerung dies für richtig erachtet. Die Zeitpunkte der Einlastung sind vollständig der Steuerung überlassen. Die Losgröße beträgt damit eins. Das **Stationsmanagement** berücksichtigt das FIFO-Prinzip innerhalb einer Station. Jede Station folgt dem Aufbau in Abbildung 2.13. Die Station arbeitet demnach mit einem Puffer am Eingang vor der Bearbeitung und mit einem Puffer am Ausgang nach der Bearbeitung. Produkte verlassen die Station somit in derselben Reihenfolge, in der sie dort ankommen. Rüstzeiten fallen nicht an. Das **Auftragsrouting** wählt aus alternativen Produktrouten. Diese ergeben sich aus dem in Abschnitt 2.1.2 beschriebenen Fähigkeitenabgleich zwischen Fähigkeitenbedarf eines Produktionsvorgangs und Fähigkeitenangebot der Stationen, also der Operationsflexibilität. Die Reihenfolgeflexibilität der Vorgänge wird hingegen vorerst vernachlässigt. Damit hat jeder Produktionsauftrag J_i lediglich einen Prozessplan P_{i1}. Im **Flottenmanagement** erfolgt die Zuweisung von WT und Produkt zu FTF bei Auftragseinlastung. WT, Produkt und zugewiesenes FTF bewegen sich anschließend gemeinsam durch die MP. Hierfür braucht es genauso viele FTFs wie WTs, es gilt: $n_{\mathrm{WT}} = n_V$. An einer Station nutzt das FTF die Parkzone, um das ihm zugewiesene Produkt auf dem WT nach der Bearbeitung wieder aufzunehmen. Das Straßennetz der MP besteht aus Zweibahnstraßen. Blockade-Situationen werden damit minimiert. Fahrzeuge können sich auf dem Wegenetz nicht überholen. Ein FTF wählt immer die kürzeste Route, welche mittels A*-Algorithmus berechnet wird. Bei vollem EP wartet ein FTF mit WT und Produkt vor der Station. Das **Störungsmanagement** befasst sich ausschließlich mit Stationsstörungen. FTFs weisen hingegen eine Verfügbarkeit von 100% auf. Bearbeitungs- und Transportzeiten sind deterministisch.

Zusammenfassend lassen sich die Anforderungen an die Entwicklung realistischer Steuerungsalgorithmen in Tabelle 2.6 stellen. Die entwickelten Entscheidungsstrategien für die fünf Entscheidungskategorien (1) sind darauf auszulegen, den Produktionsdurchsatz D_m zu maximieren, was bei gegebenem Produktmix einer Minimierung der Produktionsspanne gleicht (2). Für eine in der Realität umsetzbare Steuerung sind die Echtzeitfähigkeit der Steuerung (3) sowie die Datenverfügbarkeit der für Steuerungsentscheidungen notwendigen Produktionsdaten (4) sicherzustellen. Als Echtzeitfähigkeit definiert diese Arbeit die Bereitstellung aller notwendigen Entscheidungsinformationen in kleiner als $T_{max} = 1s$, sodass der Produktionsfluss nicht für die Entscheidungsfindung unterbrochen werden muss. Diese Definition

orientiert sich an der Umsetzung der realen E-Achsproduktion, an welche sich die vorliegende Arbeit anlehnt[13]. Die zur Entscheidungsfindung benötigte Rechenzeit hängt signifikant von der verwendeten Rechenleistung und Implementierung ab. Unter diesen Gesichtspunkten orientieren sich die Rechenressourcen der späteren Implementierung an der realen Hardware, die in der E-Achsproduktion verwendet wird. Die genannten Zusammenhänge sind auch bei der späteren Analyse der Echtzeitfähigkeit unterschiedlicher Algorithmuskonzepte zu berücksichtigen. Für einen realistischen Datenaustausch sind außerdem die steuerungsrelevanten Produktionsentitäten auf unterschiedlichen Recheneinheiten zu implementieren. Abschließend sind die genannten Anforderungen an die Steuerungsentscheidungen für unterschiedliche Produktionskonfigurationen zu erfüllen (5), was durch den Evaluationsumfang in Kapitel 6 berücksichtigt und in dieser Arbeit als Generalisierbarkeit bezeichnet wird. Dies ermöglicht die Untersuchung der Steuerungsalgorithmen unter variierenden und realistischen Randbedingungen, damit die dritte Forschungsfrage beantwortet werden kann.

Tabelle 2.6: Anforderungen an die zu entwickelnden Steuerungsalgorithmen.

Nr.	Anforderung
1	**Entscheidungen**: Lösen der fünf Entscheidungskategorien mit den in diesem Abschnitt getroffenen Annahmen: Auftragseinlastung (1.1), Stationsmanagement (1.2), Auftragsrouting (1.3), Flottenmanagement (1.4) und Störungsmanagement (1.5).
2	**Lösungsgüte**: Fokus der Steuerung auf Durchsatzmaximierung/Minimierung Produktionsspanne für einen gegebenen Produktmix.
3	**Echtzeitfähigkeit**: Berechnung aller Steuerungsentscheidungen in weniger als $T_{max} = 1s$, sodass die laufende Produktion von der Rechenzeit unbeeinflusst bleibt.
4	**Datenverfügbarkeit**: Austausch und Speichern der für Steuerungsentscheidungen notwendigen Produktionsdaten mithilfe der Kommunikation zwischen auf unterschiedlichen Recheneinheiten implementierten Produktionsentitäten.
5	**Generalisierbarkeit**: Umgang mit unterschiedlichen Produktionskonfigurationen der in diesem Abschnitt zusammengefassten MP.

Zur Erfüllung dieser Anforderungen orientiert sich die Implementierung der zentralen, dezentralen und lernenden Steuerung an der in Abschnitt 2.3.4 vorgestellten VIBN.

[13]Die Definition der Echtzeitfähigkeit ist abhängig von der Anwendung [194]. Da diese Arbeit mit Bearbeitungszeiten im Sekundenbereich arbeitet, setzt sie die Schwelle der Echtzeitfähigkeit auf $1s$. Nehmen Steuerungsentscheidungen mehr Zeit in Anspruch, summieren sich die Verlustzeiten auf, was Einfluss auf die logistischen Kenngrößen der Produktion hat.

3 Zentraler Ansatz: Rollierendes Scheduling

Dieses Kapitel erarbeitet einen zentralen Ansatz zur Steuerung der MP durch das Prinzip des rollierenden Schedulings und leistet damit einen Beitrag zur zweiten Forschungsfrage aus Abschnitt 1.2. Zuerst betrachtet Abschnitt 3.1 den bestehenden Stand der Forschung zentraler Steuerungsansätze und schlussfolgert ein zentral organisiertes Grobkonzept. Anschließend erläutert Abschnitt 3.2 die verwendete Kernmethode zur Entscheidungsfindung im zentralen Steuerungskonzept. Abschnitt 3.3 beschreibt das entwickelte Feinkonzept des rollierenden Schedulings, bevor Abschnitt 3.4 den Ansatz zusammenfasst und mit den Anforderungen aus Abschnitt 2.4.2 abgleicht.

3.1 Stand der Forschung zentraler Steuerungsansätze

Dieser Abschnitt fasst den Stand der Forschung für zentrale Steuerungsansätze in Bezug auf die MP zusammen. Abschnitt 3.1.1 verschafft hierzu einen Überblick zu den in der Literatur verwendeten Lösungsmethoden. Abschnitt 3.1.2 analysiert anschließend ausgewählte zentrale Steuerungsansätze. Abschließend fasst Abschnitt 3.1.3 die untersuchten Ansätze zusammen, ordnet sie in die Arbeit ein und zieht Schlussfolgerungen für den in der Folge entwickelten zentralen Steuerungsansatz.

3.1.1 Überblick zu zentralen Steuerungsansätzen

Abschnitt 2.2.3 führt reaktive, prädiktiv-reaktive, robust-proaktive, rollierende und MAS-Ansätze für dynamisches Scheduling ein. In klassischen zentral gelösten Scheduling-Problemen finden die ersten vier dieser Lösungsfelder Anwendung, wobei reaktive Ansätze oft als dezentral gesehen werden [195]. Die vorliegende Arbeit stuft reaktive Ansätze jedoch als zentral ein, wenn die Prioritätsregeln zentral verwaltet werden. MAS-Ansätze sind hingegen dezentral organisiert. Nach Mohan et al. [100] fokussiert sich die Literatur auf reaktive und prädiktiv-reaktive Ansätze, wobei reaktive Ansätze kurzsichtig aber flexibel agieren und prädiktiv-reaktive Ansätze durch die Vorausschau eine höhere Lösungsgüte erzielen. Robust-proaktive Ansätze finden weniger Aufmerksamkeit. Rollierende Ansätze können auch

als prädiktiv-reaktiv interpretiert werden, wenn statische Schedules bei Störereignissen neu berechnet werden [75].

Der Literatur zufolge werden in den vier Lösungsfeldern des dynamischen Schedulings unterschiedliche Lösungstechniken untersucht. Prioritätsregeln stellen die traditionellste Methode dar. Sie zählen zu den reaktiven Ansätzen und gelten als Heuristiken. Mithilfe der Regeln entscheidet sich die Ressourcenallokation [100]. Aufgrund der hohen Komplexität in Halbleiterfertigungen finden die recheneffizienten Prioritätsregeln dort häufig Anwendung. Anhand der Regeln wird beispielsweise entschieden, welcher Produktionsauftrag aus einem Maschinenpuffer als Nächstes zur Bearbeitung ausgewählt wird [21]. Ein Beispiel einer Prioritätsregel ist „First Come First Served“ (FCFS), wobei der Auftrag aus dem Puffer zur Bearbeitung gewählt wird, der zuerst im Puffer ankam [90, S. 376].

Für die globale Schedule-Berechnung in prädiktiv-reaktiven, robust-proaktiven und rollierenden Ansätzen setzen die veröffentlichten Arbeiten in einigen Fällen Metaheuristiken ein [75], [100], [117]. Hierzu zählen beispielsweise GAs, Simulated Annealing oder Tabu Search [196]. Vor allem GAs zeigen sich in vielen Veröffentlichungen als vielversprechende Methode, nahe-optimale Schedules zu berechnen [117], [118]. Außerdem werden Methoden wie Fuzzylogik, Expertensysteme, NNs, ganzzahlige Programme sowie das hybride Zusammenspiel mehrerer Techniken untersucht. Einen Überblick geben u.a. [75], [99]–[101], [117], [197]–[202].

3.1.2 Analyse ausgewählter zentraler Steuerungsansätze

Die Analyse der bestehenden Literatur zentraler Steuerungsansätze erfolgt anhand 10 ausgewählter und dem Steuerungsproblem der MP verwandter Steuerungslösungen. Die Auswahl orientiert sich an der Anwendbarkeit und Relevanz für die MP sowie an der Ausführlichkeit der Beschreibung der jeweiligen Lösungen. Die gewählten Ansätze sind den Lösungsfeldern reaktiv, prädiktiv-reaktiv, robust-proaktiv und rollierend zuzuordnen.

Reaktives Scheduling: Reaktive Ansätze wählen beispielsweise Nie et al. [203] und Heger [204], jeweils für eine der MP verwandte Werkstattfertigung bzw. ein flexibles Fertigungssystem. Der Fokus der beiden Arbeiten liegt auf der individuellen Anwendung von Prioritätsregeln in Abhängigkeit der aktuellen Produktionssituation.

Bezogen auf die MP adressieren Nie et al. [203] im Wesentlichen die Entscheidungskategorien Stationsmanagement und Auftragsrouting. An Stationen entscheiden die Prioritätsregeln, welcher Auftrag als Nächstes zur Bearbeitung geladen wird. Für eingelastete Aufträge entscheiden die Regeln hingegen die Ziel-Maschine. Maschinenstörungen und Transportzeiten berücksichtigen die Autoren nicht. Die Auftragseinlastung erfolgt exponentiell verteilt, woraus Dynamik in der Produktion resultiert. Da klassische Prioritätsregeln meist nur für spezifische Produktionssituationen gute Entscheidungen treffen, konstruieren Nie et al. allgemeingültigere Regeln mithilfe einer Kombination aus Simulation und Gene Expression

Programming, einem evolutionären Algorithmus. Die damit zentral abgeleiteten und zur Verfügung gestellten Prioritätsregeln führen zu einer höheren Performanz im Vergleich zu früheren Ansätzen und klassischen Prioritätsregeln. Aufgrund der fehlenden Berücksichtigung von Stationsstörungen, Pufferrestriktionen und Rechenzeiten fehlt dem vorgestellten Ansatz jedoch die für die Untersuchungen dieser Arbeit notwendige Realitätsnähe.

Der reaktive Steuerungsansatz Hegers [204] schließt eine ähnliche Lücke wie Nie et al. [203] in ihrer Publikation. Statt neu konstruierter Prioritätsregeln arbeitet Hegers Steuerung mit einer offline erarbeiteten zentralen Wissensbasis, welche für verschiedene Produktionssituationen dynamisch die besten Regeln und Regelparameter vorschlägt. Die Erarbeitung der Wissensbasis basiert auf einer Simulation und Regression mittels Gaußscher Prozesse. Hegers Steuerung übertrifft die Performanz von Standard-Prioritätsregeln in Werkstattfertigungssystemen signifikant. Hierbei werden allerdings Auftragsverspätungen minimiert und nicht der Durchsatz maximiert. Die situativ angewendeten Prioritätsregeln lösen laut Hegers Problemstellung das Stationsmanagement und adressieren in Ansätzen das Störungsmanagement, weil Aufträge stochastisch in das Produktionssystem eingelastet werden, wodurch Dynamik entsteht. Die tatsächliche Steuerung der Auftragseinlastung sowie das Auftragsrouting und Flottenmanagement spielen für die Steuerung hingegen keine Rolle. Die fehlende Nähe zum Steuerungsproblem dieser Arbeit erschwert die direkte Anwendung Hegers Ansatzes für die MP.

Prädiktiv-reaktives Scheduling: Dynamisches Scheduling mittels prädiktiv-reaktiver Steuerung ist das am häufigsten verwendete Lösungsfeld [75], [99]–[101]. Dies zeigt sich auch im Übergewicht prädiktiv-reaktiver Ansätze dieses Abschnitts. Chryssolouris und Subramaniam [205], van Brackel [206], Niehues et al. [207]–[209] und Cao et al. [210] legen ihren Fokus auf dieses Lösungsfeld. Sie steuern flexible Werkstattfertigungssysteme.

Chryssolouris' und Subramaniams [205] GA-basierter Ansatz trifft Entscheidungen in den Bereichen Stationsmanagement, Auftragsrouting und Störungsmanagement. Transporte und Auftragseinlastung werden nicht gesteuert. Der verwendete GA koordiniert die Werkstattfertigung, indem er bei den Produktionsereignissen Auftragseinlastung, Maschinenausfall und Maschinenreparatur jeweils einen neuen globalen Schedule für alle sich im Produktionssystem befindlichen Aufträge erstellt. Damit erzielt die GA-Steuerung signifikante Verbesserungen zu klassischen Prioritätsregeln. Die durchschnittliche Rechenzeit des GAs in [205] beträgt unter Berücksichtigung der verwendeten Rechenressourcen ca. $1000ms$. Für komplexere Problemstellungen könnte die Rechenzeit zu einem limitierenden Faktor bzgl. der Echtzeitfähigkeit des Ansatzes werden. In Summe passt der dargestellte Ansatz jedoch konzeptionell gut zur Problemstellung dieser Arbeit. Die ereignisgesteuerte Schedule-Berechnung findet deshalb auch im später entwickelten zentralen Steuerungsansatz Anwendung. Für die Schwächen in den Bereichen Flottenmanagement und Auftragseinlastung sowie für problemspezifische Anpassungen sind jedoch einige erweiternde Konzeptelemente zu implementieren.

Van Brackels [206] prädiktiv-reaktiver Steuerungsansatz adressiert die Entscheidungskategorien Stationsmanagement, Auftragsrouting und Störungsmanagement. Ein FTS wird nur am Rande berücksichtigt. Die Auftragseinlastung spielt ebenfalls keine Rolle. Mit der Komplexität des kombinatorischen Optimierungsproblems geht van Brackel im Wesentlichen durch eine zeitliche Begrenzung der Vorausschau (begrenzter Schedule-Horizont) und eine iterativ parallel zur Produktion stattfindende Schedule-Berechnung um. Der optimierte Schedule geht jeweils zum prädizierten Zeitpunkt in die Produktion über. Im Störungsfall wird in Echtzeit ein neuer gültiger, aber ineffizienter Plan unter Verwendung von Prioritätsregeln berechnet, der später wiederum durch einen optimierten Schedule ersetzt wird. Dieses komplexe Konstrukt aus Schedule-Berechnungen führt zu instabilem Steuerungsverhalten durch häufige Schedule-Neuberechnungen. Trotz der zentralen und globalen Sichtweise schneidet van Brackels Ansatz um ca. 5% schlechter ab als eine als „allwissender Gegner“ bezeichnete Benchmark-Lösung. Als wesentlichen Einflussfaktor für die Lösungsgüte nennt van Brackel den Vorausschauhorizont der Steuerung, was die vorliegende Arbeit für die zentrale Steuerungsentwicklung berücksichtigt.

Die prädiktiv-reaktive Steuerung von Niehues et al. in [207], [208] und [209] trifft Entscheidungen in den Bereichen Stations- und Störungsmanagement. Die Auftragseinlastung wird teilweise in Betracht gezogen, wohingegen Auftragsrouting und Flottenmanagement keine Rolle einnehmen. Die zentrale Steuerung zielt wie auch diese Arbeit auf gesamtheitlich optimierte Lösungen ab und hat gleichzeitig einen möglichst schlanken Umgang mit der Problemkomplexität zum Ziel. Hierzu folgt das Produktionssystem einem prädiktiv vor Produktionsstart erstellten globalen Schedule. Dieser regelt die Auftragseinlastung nach Plan und wird kontinuierlich überwacht sowie reaktiv an Störfaktoren wie Maschinenausfälle angepasst. Die reaktive Schedule-Anpassung erfolgt je nach Schwere der Planabweichungen lokal durch Prioritätsregeln oder global durch eine Neuberechnung mittels GA. Die Rechenzeit des GAs liegt unter Berücksichtigung der verwendeten Rechenressourcen hierbei nicht im für diese Arbeit definierten Echtzeitbereich von $T_{max} = 1s$. In [207] liegen die Rechenzeiten bei mehr als $10s$. In [208, S. 66] beschreibt Niehues außerdem eine Anforderung an die Entscheidungsfindung innerhalb weniger Minuten. Insgesamt schlussfolgern die Autoren im mit 81 Maschinen vergleichsweise umfangreichen Evaluationsbeispiel eine Steigerung der Termintreue und Wirtschaftlichkeit in der Werkstattfertigung. Ein direkter Vergleich zu alternativen Steuerungen bleibt jedoch aus. Bei geringeren Problemgrößen könnte die Schedule-Berechnung mittels GA in diesem Ansatz in Abhängigkeit der Rechenressourcen im Echtzeitbereich von $T_{max} = 1s$ möglich sein, weshalb die vorgestellte Implementierung als Anregung dient.

Einen weiteren prädiktiv-reaktiven Steuerungsansatz entwickeln Cao et al. [210]. Sie lösen Stationsmanagement, Auftragsrouting und in Ansätzen Störungsmanagement, weil stochastisch verteilt eingelastete Aufträge Produktionsdynamik erzeugen. Eine FTF-Flotte wird nicht berücksichtigt. Bei Ankunft eines neuen Produktionsauftrags berechnet ein sogenann-

ter „A-HEFT“ („Adaptive Heterogeneous Earliest Finish Time“)-Algorithmus in Echtzeit einen neuen Schedule für alle ausstehenden Operationen. Minimiert wird die Produktionsspanne aller Aufträge. Mittels dieser Echtzeitberechnungen erreichen Cao et al. im Vergleich zu Implementierungen von Birgin et al. [211] und Wang et al. [212] bessere Ergebnisse bei der Optimierung der Produktionsspanne. Die fehlende Berücksichtigung einer Fahrzeugflotte kombiniert mit der Fokussierung auf Montageumfänge, bei denen zwei Produktionsaufträge gleichzeitig zur selben Maschine gebracht werden müssen, machen das vorgestellte Steuerungskonzept in der Gesamtheit zu einer zu spezifischen Steuerungslösung für die Problembeschreibung dieser Arbeit. Hingegen findet das Auslösen einer Schedule-Berechnung bei Auftragseinlastung auch im später entwickelten Ansatz Anwendung.

Robust-proaktives Scheduling: Robust-proaktive Ansätze sind weniger verbreitet als reaktive und prädiktiv-reaktive. Zwei Beispiele sind die Arbeiten von Al-Hinai und ElMekkawy [213] sowie von Echsler Minguillon und Lanza [54], [55]. Die Kernkomponente der beiden Ansätze ist ein zentral erstellter robuster Schedule zur Kompensation von auftretenden Maschinenstörungen. Ob die dokumentierten robust-proaktiven Ansätze echtzeitfähig sind, ist nicht ersichtlich. Da der robuste Schedule im Voraus berechnet wird und die Adaptionen beim Rescheduling gering sein sollten, wird die Echtzeitfähigkeit in Abschnitt 3.1.3 als „teilweise berücksichtigt“ eingestuft.

Al-Hinai und ElMekkawy [213] steuern ein flexibles Werkstattfertigungssystem und adressieren Stationsmanagement, Auftragsrouting und Störungsmanagement. Mithilfe eines zweistufigen GAs suchen sie nach Schedules, die statistisch verteilte Störereignisse direkt in die Schedule-Erstellung integrieren. Der GA versucht im Bereich der erwarteten Störung, durch Pufferzeiten ein effizientes Rescheduling zu ermöglichen. Im Störungsfall werden dann trotz der Pufferzeiten betroffene Produktionsvorgänge um die Zeitdauer der Störung in die Zukunft verschoben. Die eingeplante Störverteilung basiert auf historischen Daten. Die erreichte Produktionsspanne ist stark abhängig von den in die Schedule-Erstellung eingebundenen Störungsdaten. Entsprechen die Störungen den erwarteten Verteilungen, erzielt Al-Hinais und ElMekkawys Ansatz gute Ergebnisse. Da das Störverhalten von Maschinen in der Realität häufig wenig vorhersehbar ist, kann die eingebaute Robustheit jedoch nur selten genutzt werden.

Echsler Minguillon und Lanza [54], [55] steuern eine an Greschke [16] angelehnte Matrixproduktion. Der vor der Produktion anhand eines linearen Programms berechnete robuste Schedule beinhaltet Entscheidungen zu Stationsmanagement, Auftragsrouting und Störungsmanagement. Wie bei Al-Hinai und ElMekkawy [213] strebt die Optimierung eine robuste Lösung unter Berücksichtigung der erreichten Produktionsspanne an. Der Optimierungsalgorithmus integriert deshalb für potenziell auftretende Störfaktoren zeitliche Puffer. Für von Störungen ausgelöste Unstimmigkeiten im zentralen Schedule implementieren Echsler Minguillon und Lanza zusätzlich einen dezentralen RL-Mechanismus, der angelernt wird, um

den zentralen Schedule aus Stationssicht unter Nutzung der zeitlichen Puffer partiell zu reparieren. Damit stellt der dargestellte Ansatz eine hybride Lösung aus zentral, dezentral und lernend dar. Aufgrund der dominanten zentralen Komponente wird er jedoch den zentralen Ansätzen zugeordnet. Die Autoren schlussfolgern größere Robustheit und geringere Produktionsspannen für steigende Puffergrößen. Jedoch gibt die Beschreibung der Experimente wenig Aufschluss über den Evaluationsumfang. Eine Betrachtung von Auftragseinlastung und Flottenmanagement bleibt aus.

Rollierendes Scheduling: Rollierende Scheduling-Ansätze können, wie in Abschnitt 2.2.3 beschrieben, teilweise auch als prädiktiv-reaktiv eingestuft werden. Hierbei wird ein prädiktiver Schedule reaktiv immer wieder neu berechnet. Einen derartigen Ansatz entwickeln sowohl Fang und Xi [214] als auch Sabuncuoglu und Karabuk [215], jeweils für ein flexibles Fertigungssystem.

Fangs und Xis [214] Steuerung stößt Schedule-Neuberechnungen zeitlich periodisch sowie bei Störereignissen wie Maschinenstörungen an. Ein GA berechnet die Schedules für Auftragseinlastung, Stationsmanagement und Auftragsrouting. Im Vergleich zu einer rein statischen Schedule-Berechnung reagiert der Algorithmus auf auftretende Ereignisse und plant die Produktion unter Nutzung der gegebenen Flexibilität um (Störungsmanagement). Damit verbessert sich die Produktionsspanne. Ein Vergleich zu alternativen dynamischen Scheduling-Ansätzen bleibt aus. Außerdem wird die Echtzeitfähigkeit, welche für die realistische Steuerung der MP relevant ist, zwar angegeben aber nicht nachgewiesen.

Mit einer zeitlich rollierenden Neuberechnung erzielen Sabuncuoglu und Karabuk [215] im Vergleich zu Prioritätsregeln signifikant kürzere Produktionsspannen. Die Lösungsgüte des rollierenden Ansatzes wird für 10 zufällig erstellte Probleme, bestehend aus 25 Produktionsaufträgen und sechs Maschinen, nachgewiesen. Dabei berücksichtigen die Autoren Stationsmanagement, Auftragsrouting und Flottenmanagement als Teil der Schedule-Berechnung sowie Störungsmanagement durch die Neuberechnungen zur Reaktion auf Stationsstörungen. Die Auftragseinlastung folgt anscheinend dem initial erstellten Schedule. Die Berechnungszeit der zentralen Schedules mithilfe eines „filtered beam search“-Algorithmus liegt für das Produktionsproblem unter Berücksichtigung der verwendeten Rechenressourcen bei über $100s$. Hierbei ist allerdings nicht genau ersichtlich, ob es sich um die Rechenzeit für die Berechnung eines Schedules oder um die gesamte Simulationszeit und damit mehrere Schedule-Berechnungen handelt. Echtzeitfähigkeit im Sinne dieser Arbeit scheint trotzdem nicht gegeben, da zu erwarten ist, dass die Berechnung des Schedules einen wesentlichen Anteil an der gesamten Simulationszeit hätte. Bis auf die zu hohen Rechenzeiten eignet sich das Grundkonzept dieses Ansatzes gut zur Anwendung für das MP-Steuerungsproblem, weil ein Großteil der Entscheidungskategorien adressiert wird. Deshalb nutzt der später entwickelte Steuerungsansatz einige Konzeptelemente der vorgestellten rollierenden Steuerungslösung.

3.1.3 Schlussfolgerungen für die Entwicklung der zentralen Steuerung

Tabelle 3.1 analysiert die in Abschnitt 3.1.2 beschriebenen zentralen Steuerungsansätze abschließend anhand der Anforderungen der MP (vgl. Abschnitt 2.4.2) an die fünf Entscheidungskategorien Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement (1.1-1.5) sowie anhand der Anforderungen an die Durchsatzmaximierung bzw. Minimierung der Produktionsspanne (2), die Echtzeitfähigkeit (3), die Datenverfügbarkeit (4) und die Generalisierbarkeit (5) der Ansätze.

Tabelle 3.1: Überblick zu ausgewählten zentralen Steuerungsmethoden in mit der MP verwandten Produktionssystemarten.

Legende: ● berücksichtigt ◐ teilweise berücksichtigt ○ nicht berücksichtigt, nicht ersichtlich	1.1 **Auftragseinlastung**	1.2 **Stationsmanagement**	1.3 **Auftragsrouting**	1.4 **Flottenmanagement**	1.5 **Störungsmanagement**	2 **Durchsatzmaximierung**	3 **Echtzeitfähigkeit**	4 **Datenverfügbarkeit**	5 **Generalisierbarkeit**
Reaktives Scheduling									
Nie et al. (2013)	○	●	●	○	◐	●	○	○	◐
Heger (2014)	○	●	○	○	◐	○	●	○	◐
Prädiktiv-reaktives Scheduling									
Chryssolouris und Subramaniam (2001)	○	●	●	○	●	○	●	○	◐
Van Brackel (2009)	○	●	●	◐	●	○	●	○	●
Niehues et al. (2015, 2018), Niehues (2016)	◐	●	○	○	●	○	○	◐	◐
Cao et al. (2019)	○	●	●	○	◐	●	●	○	●
Robust-proaktives Scheduling									
Al-Hinai und ElMekkawy (2011)	○	●	●	○	●	●	◐	○	●
Echsler Minguillon und Lanza (2017, 2019)	○	●	●	○	●	●	◐	○	○
Rollierendes Scheduling									
Fang und Xi (1997)	◐	●	●	○	●	●	◐	○	○
Sabuncuoglu und Karabuk (1999)	◐	●	●	●	●	●	○	○	◐

Ein Großteil der in der Tabelle gelisteten Ansätze zielt auf eine Durchsatzoptimierung bzw. die Minimierung der Produktionsspanne aller Aufträge ab. Andere Ansätze optimieren Parameter wie die Termintreue der Produktionsaufträge. Die in der Realität einer Produktion für die Steuerung relevante Datenverfügbarkeit mittels kommunizierender Produktionsentitäten wird, bis auf ansatzweise in den Arbeiten von Niehues et al., nicht berücksichtigt.

Hierzu kann das Konzept der VIBN aus Abschnitt 2.3.4 Anwendung finden. Die Steuerung der Auftragseinlastung spielt ebenso kaum eine Rolle. Meist simulieren die Autoren mit stochastisch verteilten Ankunftszeiten der Produktionsaufträge. Für die spätere Konzeptionierung der Auftragseinlastung können Löddings Kriterien für Einlastungszeitpunkte aus Abschnitt 2.2.1 dienen. Mit dem Stationsmanagement befassen sich hingegen alle Ansätze. Auch Auftragsrouting und Störungsmanagement finden vermehrt Berücksichtigung. Ein weiteres Defizit ist beim Flottenmanagement zu erkennen. Hier findet in den meisten Fällen eine Idealisierung statt oder Fahrzeuge sind für die Steuerung irrelevant. Beispielsweise spielte eine Fahrzeugflotte in Werkstattfertigungssystemen in der Vergangenheit lediglich eine untergeordnete Rolle. Zur Orientierung für die Konzeptionierung des zentralen Steuerungsansatzes dienen allerdings die in Abschnitt 2.2.3 dokumentierten FTF-Scheduling-Lösungen. Die Anforderung an Echtzeitfähigkeit, die für die reale Produktionssteuerung der MP von entscheidender Bedeutung ist, wird lediglich von Sabuncuoglu und Karabuk, von Niehues sowie von Nie et al. nicht erfüllt. Eine verhältnismäßig umfangreiche Evaluationsstudie führen van Brackel, Cao et al. sowie Al-Hinai und ElMekkawy durch. Insgesamt erschließt keiner der Ansätze die in dieser Arbeit untersuchte Breite des Steuerungsproblems und erfüllt alle Anforderungen.

Den Anforderungen am nächsten kommen die Arbeiten von van Brackel [206] sowie Sabuncuoglu und Karabuk [215]. Als prädiktiv-reaktive bzw. rollierende Ansätze erhalten sie einen globalen Blick auf die zu treffenden Entscheidungen und gehen gleichzeitig reaktiv mit den Störungen um. Die Arbeit von Sabuncuoglu und Karabuk adressiert alle fünf Entscheidungskategorien mit zeitlich periodischer Schedule-Berechnung zumindest zum Teil. Damit bildet das Konzept in [215] eine gute Ausgangsbasis für die Entwicklung eines zentralen Ansatzes zur ganzheitlichen Lösung des Scheduling-Problems einer MP. Ergänzt wird das Konzept durch Chryssolouris' und Subramaniams [205] Prinzip der ereignisgesteuerten Neuberechnung von Schedules sowie der Anwendung eines GAs zum Umgang mit Sabuncuoglus und Karabuks Rechenzeitproblem. Wie Abschnitt 3.1.1 zeigt, gelten GAs als etablierte Methode im Scheduling-Bereich. Sie bieten eine nahe-optimale Lösungsgüte und ermöglichen je nach Implementierung Echtzeitfähigkeit. Auch Niehues [208], Al-Hinai und ElMekkawy [213] sowie Fang und Xi [214] setzen erfolgreich einen GA für globales Scheduling ein. Vervollständigt wird das in Abschnitt 3.3 entwickelte zentrale Steuerungskonzept mit van Brackels [206] Idee des begrenzten Vorausschauhorizonts bei der rollierenden Schedule-Berechnung. Dies reduziert die Komplexität des rollierenden Scheduling-Problems und ermöglicht Echtzeitberechnungen mithilfe des GAs.

Bevor Abschnitt 3.3 das entwickelte Gesamtkonzept einer zentralen Steuerung erläutert, geht Abschnitt 3.2 auf den GA als Kernmethode des Konzepts ein.

3.2 Methode: Genetische Algorithmen

GAs gehen auf Hollands [216] Arbeiten in den 70er Jahren zurück. Sie eignen sich für kombinatorische Optimierungsprobleme [119, S. 2]. Sie zählen zur Klasse der populationsbasierten Metaheuristiken, welche nach Talbi [196, S. 1] Anwendung für Optimierungsprobleme hoher Komplexität finden, die bei der häufigen NP-Vollständigkeit von Scheduling-Problemen zutrifft. GAs zeichnen sich durch ihre Effizienz mittels parallelisierter Lösungssuche aus und erarbeiten damit gute Näherungslösungen für komplexe Probleme [217, S. 59], [196, S. 25]. Die parallele Lösungssuche mithilfe einer Lösungsschar sowie die zufallsgeführte Suche im Lösungsraum vermeiden hierbei lokale Optima. In der Punkt-zu-Punkt-Suche klassischer Gradientenverfahren stellt dies ein Risiko dar [218, S. 217]. Als Vertreter evolutionärer Algorithmen kommt die Motivation für GAs aus der Natur. Der kombinatorische Algorithmus orientiert sich hierbei am evolutionären Prozess der Weiterentwicklung von Individuen wie Menschen, Tieren oder Pflanzen, die sich Generation für Generation immer besser an die Umweltbedingungen anpassen, wobei nicht angepasste Individuen aussterben [219, S. 15]. Der GA arbeitet demnach mit dem Prinzip der Veränderung des Genoms einer Spezies und mit dem Prinzip der natürlichen Auslese [218, S. 210].

Ein Individuum I repräsentiert im GA eine Lösung des Optimierungsproblems. Eine Population besteht aus mehreren Individuen und damit einer Menge an Lösungen [220, S. 159f.]. Ein Individuum einer Population wird i.d.R. als Zeichen- bzw. Zahlenkette dargestellt [119, S. 37], beispielsweise binär codiert:

$$I = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0] \tag{3.1}$$

Die Codierung in GAs ist problemspezifisch. Sie hängt außerdem wechselseitig von den Mechanismen ab, die zur Durchsuchung des Lösungsraums angewendet werden. Nicht jeder GA-Mechanismus passt zu jeder Problemcodierung und umgekehrt [218, S. 222]. Jedes Element der codierten Zahlenkette wird als Gen bezeichnet. Der Begriff Allel bezeichnet den Wert eines Gens [221, S. 34]. Der Locus beschreibt die Lage eines Gens innerhalb des Individuums [119, S. 34]. Außerdem werden die Begriffe Genotyp und Phänotyp unterschieden. Der Genotyp ist die Darstellung des Individuums als codierte Lösung, also als Zeichen- bzw. Zahlenkette. Der Phänotyp hingegen entspricht der decodierten Lösung des Optimierungsproblems, in Scheduling-Problemen dem zu optimierenden Gantt-Diagramm. Eine Decodierungsfunktion überführt den Genotyp in den Phänotyp. Damit legt sie die Repräsentation einer Lösung des Optimierungsproblems mithilfe des codierten Individuums fest. Die Güte eines Individuums wird durch die Fitnessfunktion $J(I)$ anhand des decodierten Phänotyps bestimmt. Beispielsweise erfolgt die Bewertung eines decodierten Schedules. Ziel der Optimierung ist die Maximierung der Fitness [220, S. 159ff.].

Gegeben ist also eine zu maximierende Fitnessfunktion $J(I)$. Gesucht wird das Individuum

I aus der problemspezifischen Definitionsmenge D, das diese Funktion maximiert:

$$J_{max} = \max_{I \in D} J(I), \qquad I_{max} = \arg\max_{I \in D} J(I). \tag{3.2}$$

Die evolutionäre Optimierung der Fitnessfunktion durchläuft N_{GEN} Generationen mit einer Population von N_{POP} Individuen. Alternativ endet die Optimierung bei einem festgelegten Abbruchkriterium, beispielsweise bei Erreichen einer vordefinierten Lösungsgüte. Die Größe der Population bleibt Generation für Generation konstant. Jede Iteration strebt einen Schritt in Richtung der optimalen Lösung an [196, S. 200, S. 342], [217, S. 64f.]. Die parallele Suche nach dem Optimum folgt einem vordefinierten Ablaufschema. Alg. 3.1 beschreibt das Ablaufschema mittels Pseudocode.

Algorithmus 3.1 GA in Anlehnung an [218, S. 218], [196, S. 200].

1: Initialisiere die Hyperparameter: $N_{\text{GEN}}, N_{\text{POP}}, p_R, p_M$.
2: Wähle eine **initiale Population** λ_0 der Größe N_{POP}.
3: **for** $i \leftarrow 1$ to N_{GEN} **do**
4: **Evaluiere** jedes decodierte Individuum I der Population durch Berechnung von $J(I)$.
5: **Selektiere** eine Elternpopulation auf Basis der Bewertungen $J(I)$.
6: **Generiere Kindpopulation** durch Rekombination und Mutation mit Wahrscheinlichkeiten von p_R und p_M.
7: Ersetze die Population durch die Kindpopulation: $\lambda_{i-1} \leftarrow \lambda_i$.
8: **end for**

Nach der Initialisierung der Hyperparameter wird beispielsweise zufällig eine Anfangspopulation λ_0 der Größe N_{POP} erstellt. Jedes der initialisierten Individuen entspricht hierbei einem Wert innerhalb des Definitionsbereichs: $I \in D$. In jeder Generation erfolgt dann zuerst eine Bewertung der Individuen der aktuellen Population mittels der Fitnessfunktion $J(I)$. Anhand der Bewertung wird eine Anzahl von N_{SEL} Individuen als Elternpopulation selektiert. Hierbei entspricht N_{SEL} meist N_{POP}, sodass die Populationsgröße von Generation zu Generation konstant bleibt. Der Selektionsprozess wird demnach so lange wiederholt, bis $N_{\text{SEL}} = N_{\text{POP}}$ Individuen als Eltern zur Verfügung stehen. Damit können einzelne Individuen auch mehrfach selektiert werden [218, S. 244]. Die selektierten Elternindividuen dienen als Grundlage für die Rekombinationen und Mutationen hin zu einer Kindpopulation. Zusammen mit der Selektion bilden diese Mechanismen das Kernstück der evolutionären Entwicklung. Sie durchsuchen die Lösungsmenge nach besseren Lösungen mit dem Ziel einer höheren durchschnittlichen Generations-Fitness. Durch Rekombination werden selektierte Elternindividuen nach einem definierten Schema kombiniert, wobei neue veränderte Kindindividuen erzeugt werden. Aus zwei Elternindividuen entstehen hierbei zwei Kindindividuen. Die Wahrscheinlichkeit für die Rekombination zweier Individuen wird mit p_R bezeichnet. Mutationsmechanismen modifizieren hingegen einzelne Gene eines Individuums, wodurch sich dieses einzelne Individuum ändert. Dadurch erfolgt eine enger gefasste Suche an den entspre-

chenden Stellen des Suchraums. Die Wahrscheinlichkeit, dass ein Individuum mutiert, wird mit p_M bezeichnet. Die durch die Selektion und die anschließende wahrscheinlichkeitsbasierte Rekombination und Mutation entstandene neue Kindpopulation λ_i ersetzt die vorherige λ_{i-1} und wird zur Population der nächsten Iteration. Werden einzelne selektierte Elternindividuen weder rekombiniert noch mutiert, bleiben sie unverändert in der Kindpopulation erhalten. Am Ende der Optimierung steht das fitteste Individuum I_{max} als Lösung bereit [217, S. 63-66], [218, S. 218f.], [222].

Initialisierung der Ausgangspopulation: Koch [217, S. 71f.] differenziert drei Arten zur Initialisierung der Ausgangspopulation λ_0 der Größe N_{POP}, wobei jedes Individuum der problemspezifischen Codierung entspricht:

- **Zufallsgenerierung**: Bei der Zufallsgenerierung werden Individuen randomisiert ermittelt [196, S. 194f.]. Vorteil ist die Heterogenität der Ausgangspopulation und dadurch eine breite Abdeckung des Lösungsraums. Nachteil ist, dass möglicherweise unzulässige Individuen erzeugt werden, welche entweder aussortiert oder anderweitig bearbeitet werden müssen [223, S. 239].

- **Heuristische Generierung**: Die heuristische Generierung der Ausgangspopulation garantiert das Erzeugen zulässiger Individuen, indem oben beschriebene Fälle vermieden werden. Die erzeugten Individuen liegen je nach Lösungsverfahren bereits näher an der optimalen Lösung. Nachteilig sind jedoch die Homogenität der entstehenden Population und die damit einhergehende Einschränkung des Lösungsraums [196, S. 198].

- **Seeding**: Die dritte Methode „Seeding“ kombiniert beide Ansätze und stellt damit einen Kompromiss aus den genannten Vor- und Nachteilen dar [224, S. 119].

Selektion aus der Population: Die Selektionsphase des GAs basiert auf der Bewertung der Lösungen einer Population. Die Bewertung der einzelnen Individuen hängt von der Decodierungsfunktion ab. Die Decodierung und die zugehörige Fitnessfunktion $J(I)$ sind wie die Codierung problemspezifisch. Sie werden deshalb in Abschnitt 3.3 in Bezug auf das zu lösende Scheduling-Problem beschrieben. Insgesamt gilt: Je fitter das Individuum, desto größer ist seine Überlebenswahrscheinlichkeit. Drei gängige Verfahren sind die Roulette Wheel Selection, die rangbasierte Selektion und die Wettkampfselektion [196, S. 206f.], [225, S. 266f.]:

- **Roulette Wheel Selection**: Für die Roulette Wheel Selection wird jedem Individuum eine Selektionswahrscheinlichkeit proportional zu seiner Fitness zugewiesen. Je höher der Fitnesswert, desto größer ist die Selektionswahrscheinlichkeit.

- **Rangbasierte Selektion**: Die rangbasierte Selektion sortiert die Individuen entsprechend ihrer Fitnesswerte in einer Rangfolge. Die Selektionswahrscheinlichkeiten werden

entsprechend der Ränge zugewiesen und sind unabhängig vom Absolutwert der Fitness eines Individuums.

- **Wettkampfselektion**: Die Wettkampfselektion selektiert in einzelnen Wettkämpfen jeweils das fitteste Individuum aus einer zufällig ausgewählten Anzahl an Individuen aus der Population.

Rekombination und Mutation: Sowohl für Rekombination als auch für Mutation hängt das Design der GA-Mechanismen von der Problemcodierung ab. Außerdem existieren keine Mechanismen, die für alle Problemstellungen die besten Ergebnisse liefern [218, S. 222]. Talbi [196, S. 213] spricht von binärer Rekombination, weil zwei Eltern zu zwei Kindern kombiniert werden. Mutation befasst sich hingegen mit einem einzelnen Individuum und wird deshalb nicht als binär bezeichnet [196, S. 208]. Eine erfolgreiche Rekombination berücksichtigt zwei Eigenschaften: Erstens sollte ein Kindindividuum Erbgut beider Eltern enthalten. Zweitens hat die Rekombination die Validität der Kindindividuen zum Ziel. Mutation hat hingegen drei Ziele: Erstens ist sicherzustellen, dass durch Mutation jede Lösung des Suchraums erreicht werden kann. Zweitens sollte das mutierte Individuum auch hier eine valide Lösung darstellen. Drittens ist durch Mutation nur eine minimale Veränderung der Lösung anzustreben [196, S. 208-220]. Je nach Problemstellung kann die GA-Performanz durch Kombination mehrerer Rekombinations- oder Mutationsmechanismen gesteigert werden [226].

Abbildung 3.1 visualisiert zwei gängige Rekombinationsmechanismen (engl.: crossover). Beim 1-Punkt-Crossover werden zwei Elternindividuen P_1 und P_2 an einer anhand einer Gleichverteilung zufällig bestimmten Stelle der Individuen gekreuzt. Die Kindindividuen K_1 und K_2 erhalten Erbgut beider Elternteile. Der Austausch des Erbguts findet im Bild links vom Crossover-Punkt statt [221, S. 39]. Beim 2-Punkt-Crossover werden statt einem zwei zufällig gewählte Crossover-Punkte bestimmt. Der Austausch des Erbguts findet dann zwischen den beiden Punkten statt [119, S. 88]. Einen Überblick zu Crossover-Mechanismen geben Umbarkar und Sheth [227].

$$P_1 = [0\ 1\ 1\ 0\ 1 \mid 1\ 0\ 0\ 0] \qquad P_2 = [1\ 1\ 0\ 0\ 0 \mid 1\ 1\ 1\ 0]$$

$$\xrightarrow[\text{Crossover}]{\text{1-Punkt}}$$

$$K_1 = [1\ 1\ 0\ 0\ 0 \mid 1\ 0\ 0\ 0] \qquad K_2 = [0\ 1\ 1\ 0\ 1 \mid 1\ 1\ 1\ 0]$$

$$P_1 = [0\ 1 \mid 1\ 0\ 1\ 1 \mid 0\ 0\ 0] \qquad P_2 = [1\ 1 \mid 0\ 0\ 0\ 1 \mid 1\ 1\ 0]$$

$$\xrightarrow[\text{Crossover}]{\text{2-Punkt}}$$

$$K_1 = [0\ 1 \mid 0\ 0\ 0\ 1 \mid 0\ 0\ 0] \qquad K_2 = [1\ 1 \mid 1\ 0\ 1\ 1 \mid 1\ 1\ 0]$$

Abbildung 3.1: 1-Punkt-Crossover und 2-Punkt-Crossover zweier Elternindividuen P_1 und P_2 zu zwei Kindern K_1 und K_2, Quelle: Eigene Darstellung in Anlehnung an [119, S. 40, S. 88].

Abbildung 3.2 stellt eine beispielhafte Mutation eines Elternindividuums P hin zu einem Kindindividuum K dar. Mit der Wahrscheinlichkeit p_M mutiert ein Gen des Elternchromosoms. In der im Bild gewählten Binärdarstellung erfolgt die Mutation von 1 zu 0 oder von 0 zu 1 [119, S. 41]. In nicht-binären Repräsentationen kann das Gen auch andere Werte annehmen. Außerdem existieren Mutationsmechanismen, die mit der Reihenfolge der Gene arbeiten [196, S. 209]. Einen Überblick zu weiteren Mutationen gibt Talbi [196, S. 208-213].

$$P = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]$$
$$K = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]$$

Abbildung 3.2: Mutation zweier Gene von einem Elternindividuum P zu einem Kindindividuum K, Quelle: Eigene Darstellung in Anlehnung an [119, S. 41].

Zusammenfassung: Tabelle 3.2 fasst die oben definierten Stellgrößen des GAs zusammen. Die GA-Gestaltung erfolgt größtenteils in Abschnitt 3.3. Mathematische Analysen haben

Tabelle 3.2: Zusammenfassung der Stellgrößen des GAs.

Stellgröße	**Beschreibung**
Codierung	Zeichen- bzw. Zahlenkette zur codierten Repräsentation einer Lösung
Initialisierung	Erzeugen einer Ausgangspopulation
Decodierung	Umwandlung der codierten Lösung, beispielsweise in einen Schedule
Evaluation	Ermittlung der Güte einer Lösung
Selektion	Wählen von Elternindividuen
Rekombination	Paarung zweier Elternindividuen
Mutation	Lokale Variation eines Individuums
N_{POP}	Populationsgröße
N_{SEL}	Anzahl der selektierten Elternindividuen
N_{GEN}	Anzahl der Generationen bei der Optimierung
p_R	Wahrscheinlichkeit für Rekombination zweier Individuen
p_M	Wahrscheinlichkeit für Mutation eines Individuums

gezeigt, dass die Einstellung der Stellgrößen als problemspezifisch gilt und die Generalisierungsfähigkeit auf andere Problemstellungen nicht implizit gegeben ist [218, S. 238], [228]. Zur Wahl und Einstellung der Parameter orientiert sich der in Abschnitt 3.3 entwickelte GA deshalb an verwandten Scheduling-Implementierungen aus der Literatur. Für den Vergleich

von Parameterkombinationen lassen sich Konvergenzkurven mit der evolutionären Entwicklung der durchschnittlichen Populationsfitness verwenden [218, S. 245]. Eine Untersuchung verschiedener Parameterkombinationen erfolgt in Abschnitt 6.3.1.

3.3 Konzeptionierung des rollierenden Schedulings

Abschnitt 3.1.3 leitet aus dem bestehenden Stand der Forschung ein Grobkonzept zur zentralen Steuerung der MP ab, vor allem auf Basis von [205], [206], [215]. Abschnitt 3.2 stellt anschließend GAs als Methode zur Schedule-Berechnung vor. Darauf basierend fasst Abbildung 3.3 einleitend den entwickelten zentralen Steuerungsansatz zusammen und stellt überblicksweise die Lösung der fünf Entscheidungskategorien aus Abschnitt 2.2.2 dar. Teile dieses Ansatzes wurden in [29]–[31] veröffentlicht.

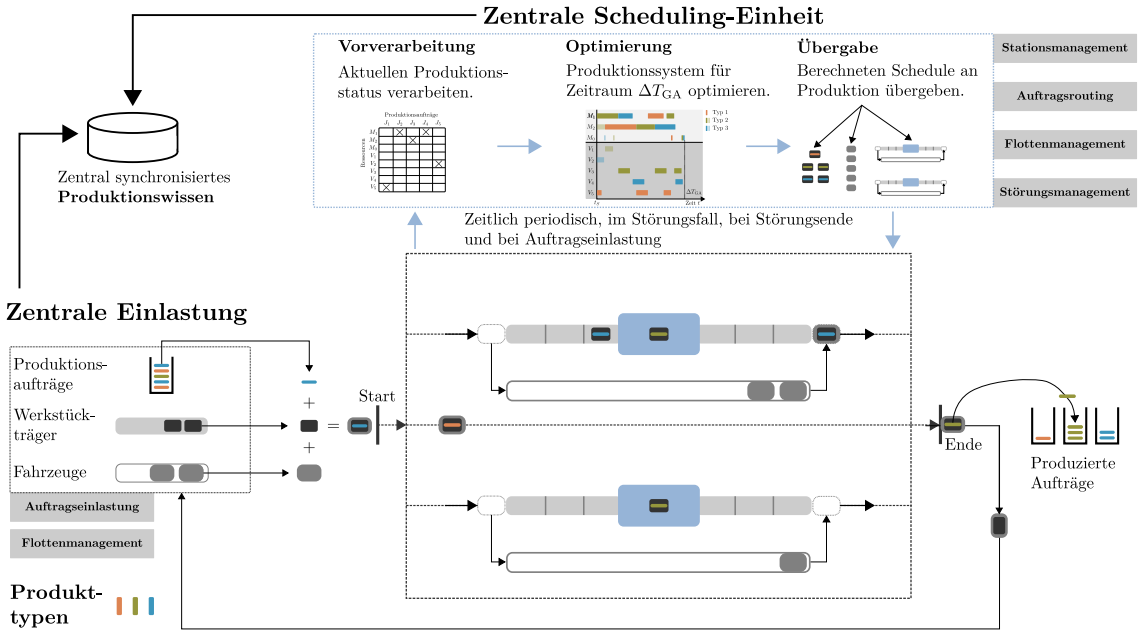


Abbildung 3.3: Rollierendes Scheduling zur Steuerung der MP: Eine zentral organisierte Auftragseinlastung ordnet Produkte zu WTs und Fahrzeugen zu und lastet die Produkte ein, während ein GA zu regelmäßigen Zeitpunkten sowie ereignisorientiert die Ressourcenallokation optimiert, um die eingelasteten Produkte durch die MP zu koordinieren.

Ein zentral organisiertes Einlastungsmodul steuert die Auftragseinlastung mittels vordefinierter Regeln und orientiert sich dabei am geplanten Produktmix für das Produktionssystem. Da einem Produkt durch die Einlastung sowohl ein WT als auch ein FTF zugewiesen wird, erfolgt auch der erste Schritt des Flottenmanagements bei der Einlastung. Die Steuerung der eingelasteten Aufträge übernimmt anschließend eine zentrale Scheduling-Einheit, deren Entscheidungen das Stationsmanagement, das Auftragsrouting, das Flottenmanagement und das Störungsmanagement betreffen. Hierzu berechnet das Scheduling-Modul, basierend auf dem aktuellen Produktionsstatus, globale Schedules für einen vordefinierten Zeit-

raum ΔT_{GA}, welche die Entscheidungen für die genannten Entscheidungskategorien beinhalten. Eine Optimierung wird hierbei zeitlich periodisch in festen Intervallabständen von Δt_s sowie ereignisorientiert bei Auftragseinlastung eines Auftrags, bei Beginn einer Stationsstörung und bei Wiederverfügbarkeit einer Station angestoßen. Im Gegensatz zum Vorausschauhorizont bei van Brackel in [206] und bei Niehues in [208] beträgt der Vorausschauhorizont ΔT_{GA} in dieser Arbeit lediglich ein einstelliges Vielfaches der Zykluszeit der MP. Dadurch liegt die Vorausschau im Minutenbereich, anstatt z.B. ganze Produktionstage zu planen. Hiermit werden Echtzeitberechnungen möglich, welche in den genannten Arbeiten nicht erreicht wurden. Der tatsächliche Vorausschauhorizont wird in Abschnitt 6.3.1 untersucht. Ein GA führt die Scheduling-Berechnungen durch. Nach jeder Optimierung übergibt der GA die getroffenen Entscheidungen an die betroffenen Produktionsentitäten, also die Produkte, Stationen und Fahrzeuge, um auf deren Basis die Produktionsabläufe weiterzuführen. Dieser iterative Scheduling-Prozess zerlegt das Steuerungsproblem in Zeitfenster, verringert dadurch die Komplexität des Optimierungsproblems und findet gesamtheitlich nahe-optimale Lösungen für die Abläufe in der MP. Der entsprechende Nachweis erfolgt in der Evaluation in Kapitel 6.

Der Ansatz stellt eine Analogie zur Bahnsteuerung bei Robotern dar. Diese berechnen die Bewegungsbahnen für mehrere Zeitschritte im Voraus (analog zu ΔT_{GA}), um einen möglichst gleichmäßigen Bahnverlauf zu erreichen. Außerdem fließen Sensordaten als Rückmeldung für die Steuerung mit ein, um beispielsweise die Trägheit der Bewegungsachsen des Roboters in Abhängigkeit von der Stellung der Achsen regelmäßig zu berücksichtigen. Die Berechnung bis zum Ende einer Bahnbewegung ist hierbei zu rechenaufwändig (vgl. bspw. [229]).

Zur Erläuterung des in Abbildung 3.3 visualisierten zentralen Steuerungskonzepts betrachtet Abschnitt 3.3.1 zuerst die Auftragseinlastung, bevor Abschnitt 3.3.2 und Abschnitt 3.3.3 auf die weiteren Entscheidungskategorien eingehen. Abschließend fasst Abschnitt 3.4 die wesentlichen Aspekte des zentralen Steuerungskonzepts zusammen und vergleicht den Ansatz mit den in Abschnitt 2.4.2 definierten Anforderungen der MP-Steuerung.

3.3.1 Auftragseinlastung

Die Auftragseinlastung geht nach den in Abschnitt 2.4.2 getroffenen Annahmen von einem vordefinierten Produktmix aus. Dieser legt die Anzahl der zu fertigenden Produkte pro Produkttyp fest und in welchem Verhältnis sie einzulasten sind. Zusammen mit der empfohlenen Periodizität der einzulastenden Produkte bildet der Produktmix das Produktionsprogramm und damit eine Liste abzuarbeitender Produktionsaufträge. Zu treffen sind Entscheidungen über die Zuweisung von WTs und FTFs sowie über die Zeitpunkte der Einlastung der einzelnen Produktionsaufträge. Eine erfolgreiche Einlastung erfordert deshalb die Verfügbarkeit eines WTs und eines FTFs, auf denen das Produkt im System gestartet werden kann. Damit betrifft die Einlastung auch einen Teil des Flottenmanagements. Die Anzahl verfügbarer

WTs bzw. FTFs ($n_{\mathrm{WT}} = n_V$, Annahme in Abschnitt 2.4.2) begrenzt den Produktionsbestand in der MP zu: $PB_{max} = n_{\mathrm{WT}} = n_V$. Hierfür bietet sich eine Orientierung an der in Abschnitt 2.2.1 vorgestellten einfachen Conwip-Steuerung an, welche einen neuen Auftrag einlastet, wenn gilt: $PB < PB_{max}$ [74, S. 373-385]. Mit diesem Hintergrund, ergänzt mit einigen problemspezifischen Erweiterungen, folgt die zentrale Auftragseinlastung dem Produktionsbestand und basiert auf zwei Prinzipien:

1. **Initiale Einlastung**: Im Initialzustand der Produktion stehen n_{WT} WTs und n_V Fahrzeuge hintereinander in der Quelle der MP bereit. Die periodische Auftragseinlastung beginnt in der vordefinierten Auftragsreihenfolge des Produktionsprogramms unter Verwendung der verfügbaren WTs und FTFs. Die WTs und FTFs werden in der vorhandenen Reihenfolge nacheinander zugewiesen. Damit weist die zentrale Einlastungseinheit die verfügbaren n_{WT} WT-FTF-Paare (WT_1/V_1, WT_2/V_2 etc.) den ersten Produktionsaufträgen des Produktionsprogramms zu und lastet sie direkt ein. Zur Verhinderung von Stausituationen im Produktionsanlauf kann ein zeitlicher Mindestabstand ΔT_{AE} zwischen zwei Einlastungszeitpunkten zweier Aufträge J_i und J_{i+1} von $S_{J_i} - S_{J_{i+1}} \geq \Delta T_{\mathrm{AE}}$ eingestellt werden.

2. **Bestandsgeregelte Einlastung**: Sind alle WTs und FTFs in der MP im Einsatz, gilt: $PB = PB_{max}$. Die zentrale Steuerungseinheit wartet dann auf ein verfügbar werdendes WT-FTF-Paar durch die Fertigstellung eines Produktes, wodurch PB unter die Bestandsgrenze PB_{max} fällt. Unter Beibehaltung des Produktionsprogramms erfolgt dann die Einlastung des nächsten anstehenden Produktionsauftrags, sobald der WT und das FTF in der Quelle der MP ankommen. So ergeben sich die Einlastungszeitpunkte S_{J_i} der Aufträge automatisiert. Durch dieses Verfahren bleiben FTF und WT anhand der initialen Zuweisung dauerhaft verheiratet.

Diese Art der Einlastung setzt das geplante Produktionsprogramm mit der Losgröße eins um. Außerdem erfolgt die feste Verknüpfung von Produktionsauftrag, WT und FTF für die abzuarbeitenden Produktionsprozesse des Produktes, was ebenfalls als Annahme in Abschnitt 2.4.2 aufgeführt ist. Die verfügbaren WTs und FTFs sind dem zweiten Prinzip der Einlastung entsprechend dauerhaft im Einsatz.

Die auf die Auftragseinlastung folgende Steuerung der Produktionsabläufe übernimmt das Stationsmanagement, das Auftragsrouting und den zweiten Teil des Flottenmanagements.

3.3.2 Stationsmanagement, Auftragsrouting und Flottenmanagement

Die aktive Einflussnahme auf eingelastete Produktionsaufträge in den Entscheidungskategorien Stationsmanagement, Auftragsrouting und Flottenmanagement erfolgt durch ein zentral

organisiertes Scheduling-System. Dieses arbeitet sowohl zeitlich periodisch als auch ereignisorientiert bei Auftragseinlastung eines Produktes sowie im Störungsfall und bei Störungsende einer Station.

Die zeitliche Zerlegung des Scheduling-Problems zeigt Abbildung 3.4. Eine Neuberechnung wird hierbei in regelmäßigen Abständen von Δt_s angestoßen. Dies sorgt für regelmäßiges Hinterfragen der aktuellen Planung auf Basis des zum jeweiligen Scheduling-Zeitpunkt t_s vorliegenden Produktionsstatus. Anstatt alle für den anstehenden Produktionstag zu produzierenden Aufträge zu berücksichtigen, schaut der Algorithmus bei jeder Schedule-Berechnung ΔT_{GA} voraus. Eine Analyse dieses Parameters erfolgt in Abschnitt 6.3.1.

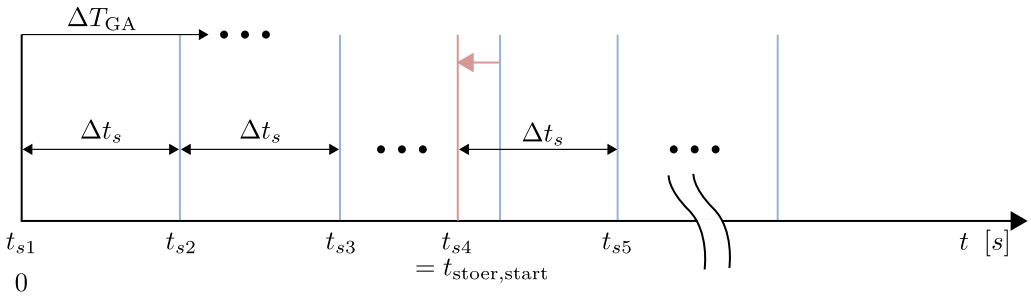


Abbildung 3.4: Zeitlich periodische sowie ereignisorientierte Berechnung von Schedules zu Zeitpunkten t_s bei einer Vorausschau von ΔT_{GA}, Quelle: Eigene Darstellung in Anlehnung an [29].

Die dargestellte zeitliche Zerlegung der Produktionszeit reduziert die Problemkomplexität und ermöglicht Echtzeitberechnungen mithilfe des GAs. Damit dieses Prinzip funktioniert, muss ΔT_{GA} so gewählt werden, dass der GA für die Periode Δt_s, in der kein weiterer Scheduling-Lauf geplant ist, genügend Entscheidungen für die MP zur Verfügung stellt. ΔT_{GA} ist hierbei kein dynamischer Parameter, der von der Produktionssituation abhängt. Stattdessen wird er einmalig vor dem Produktionsstart eingestellt. Somit muss die Wahl von ΔT_{GA} mit der Wahl von Δt_s abgestimmt sein, um alle möglichen Produktionssituationen in der MP abzudecken. Es muss gelten:

$$\Delta T_{\mathrm{GA},min} \geq \Delta t_s + UEZ_{ipl,max} \;\; \forall J_i \in J, \forall P_{ip} \in P_i, \forall O_{ipl} \in O_{ip}. \tag{3.3}$$

ΔT_{GA} schließt damit die anstehende Periode Δt_s bis zum nächsten Scheduling-Zeitpunkt ein. Zusätzlich ist die größtmögliche Übergangszeit $UEZ_{ipl,max}$ aller im Produktionsprogramm enthaltenen Produkte J einzuplanen. Sie setzt sich aus der größten auftretenden Transportzeit aller Aufträge $t_{\mathrm{TR},max}$ und den Mindest-Liegezeiten in den Puffern $t_{\mathrm{AP},min}$ und $t_{\mathrm{EP},min}$ zusammen. Bei allen drei Werten handelt es sich um theoretische Werte. Verlängern sich diese Zeiten in der realen Produktion, stellt dies kein Problem dar, weil dem GA damit lediglich mehr Zeit verschafft wird, wieder neue Entscheidungen für die Aufträge bereitzustellen. Da $UEZ_{ipl,max}$ die größte Transportzeit der zu fertigenden Aufträge J beinhaltet, ist (3.3) auf den schlechtestmöglichen Fall ausgerichtet und deckt damit alle möglichen Pro-

duktionssituationen ab. Die Übergangszeit ist zu beachten, weil der GA mit Fokus auf die Produktionsvorgänge plant und damit sichergestellt werden muss, dass die Vorausschau nicht mitten in einem Transport bzw. Übergang endet, sondern zumindest die nächste Operation noch einplant. Andernfalls könnten einem Auftrag die notwendigen Entscheidungen während der Periode Δt_s zum Ende hin fehlen. Details zu diesem Zusammenhang folgen im Verlauf dieses Abschnitts.

Neben dem zeitlich periodischen Auslösen von Schedule-Berechnungen zeigt Abbildung 3.4 auch die ereignisgesteuerte Berechnung. Zum Zeitpunkt $t_{\text{stoer,start}}$ geht eine Station der MP in Störung, was eine Neuberechnung auslöst. Dadurch verschiebt sich auch t_{s4} auf diesen Zeitpunkt und die periodische Planung startet von $t_{\text{stoer,start}}$ an von Neuem. Gleiches gilt für ein Störungsende $t_{\text{stoer,ende}}$ sowie bei Auftragseinlastung eines Auftrags J_i zum Zeitpunkt S_{J_i}. Letzteres Ereignis löst eine Schedule-Berechnung aus, weil für einen eingelasteten Auftrag noch keine Entscheidungen getroffen wurden.

Um zu Zeitpunkten t_s einen Schedule berechnen zu können, ist eine Erfassung des aktuellen Produktionsstatus aller aktiven Produktionsaufträge J_i notwendig. Diese erfolgt im Rahmen einer Vorverarbeitungsphase. Anschließend folgt die Optimierung mittels GA und die Übergabe der berechneten Entscheidungen an die Produktionsentitäten. Die nächsten drei Abschnitte gehen auf diese drei Phasen ein.

Vorverarbeitung der Produktionsdaten

Zur Verarbeitung der aktuellen Produktionsdaten benötigt die zentrale Scheduling-Einheit ein aktuelles Bild der Produktion und somit aller aktiven Aufträge J_{akt}. Jeder Auftrag wird von der zentralen Scheduling-Einheit hierfür als mathematischer Graph bzw. mathematischer Wurzelbaum verwaltet, analog zur Darstellung des Produktgraphen in Abschnitt 2.1.2. Mathematische Graphen sind hierbei eine gängige Darstellungsform im Scheduling-Bereich [71], [230], [231]. Die folgenden Absätze beschreiben erstens die mathematische Graphenrepräsentation eines Auftrags, zweitens die potenziellen Produktionsstatus eines Auftrags in Graphenform, drittens die Berechnung der verbleibenden Prozesszeiten je nach Status sowie viertens die Vorausschau um den Zeitraum ΔT_{GA} pro Auftrag.

Graphenrepräsentation: Abbildung 3.5 zeigt die mathematische Darstellung des Wurzelbaums $T_i = (U, E)$ anhand eines Beispielprodukts mit Operationsflexibilität. Der Graph[1] besteht aus der Wurzel w sowie Knoten $u \in U$ und Kanten $e \in E$. Die Wurzel repräsentiert den Produktionsstart in der Quelle der MP. Ein Knoten u repräsentiert für einen Auftrag J_i eine Operation O_{ipl} an einer definierten Station M_k aus der Teilmenge $M_{ipl} \subseteq M$ mit einer Bearbeitungszeit von t_{ipl}. Die Bearbeitungszeit zeigt sich in der auf die Zeitachse projizierten Länge des Knotens u. Der dargestellte Auftrag arbeitet mit nur einem Prozessplan P_{i1}. Er beinhaltet damit keine Reihenfolgeflexibilität (vgl. Annahme in Abschnitt 2.4.2). Die

[1] Graphentheoretische Grundlagen sind beispielsweise in [232], [233] oder [234] nachzulesen.

Operationsflexibilität des Auftrags ist beispielsweise bei Operation O_{i11} zu erkennen, welche sowohl an M_1 als auch an M_2 durchgeführt werden kann. Somit gilt: $M_{i11} := \{M_1, M_2\}$. Einen zweiten Fall von Operationsflexibilität zeigt O_{i12} mit $M_{i12} := \{M_3, M_4\}$. Eine Kante e repräsentiert hingegen den Übergang von einer Operation an einer Station M_{k1} zur nächsten Operation O_{ipl} an der Folgestation M_{k2}. Hierbei bezeichnet $UEZ_{ipl,M_{k1}\rightarrow M_{k2},min}$ die minimal mögliche Übergangszeit unter Berücksichtigung der minimalen Aufenthaltszeiten im AP ($t_{\mathrm{AP},min}$) und EP ($t_{\mathrm{EP},min}$) sowie der theoretischen Transportzeit $t_{\mathrm{TR},M_{k1}\rightarrow M_{k2}}$. Wiederum stellt die auf die Zeitachse projizierte Länge der Kante die Dauer der Übergangszeit dar.

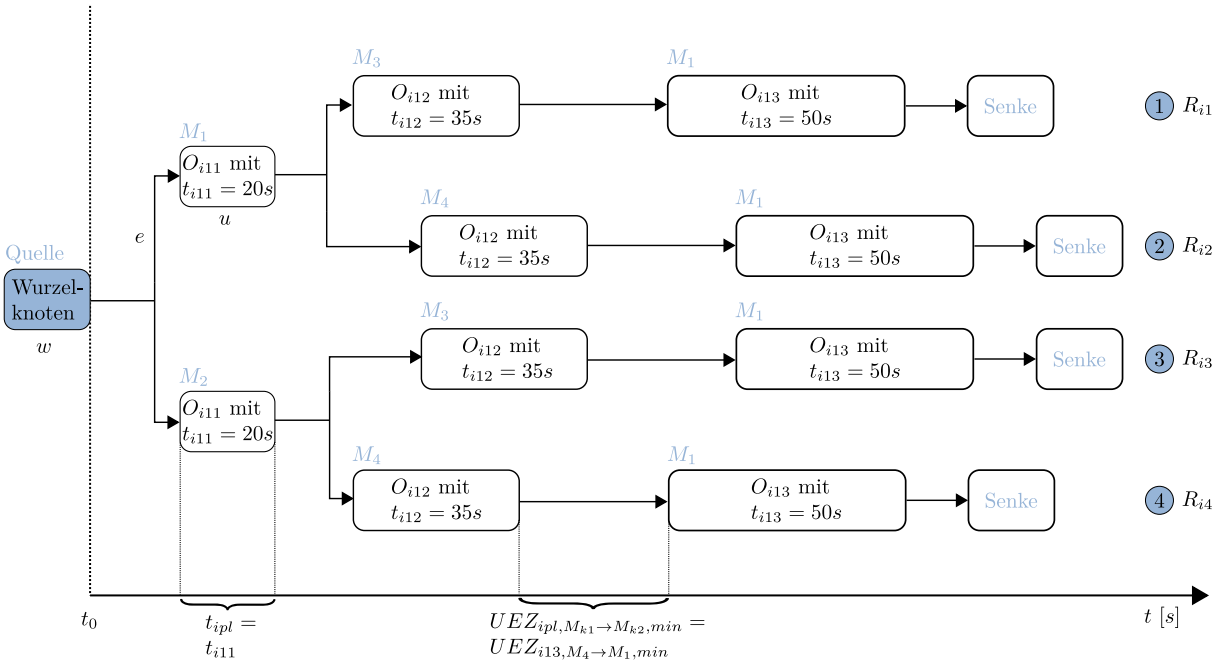


Abbildung 3.5: Repräsentation eines Produktes als mathematischer Wurzelbaum $T_i = (U, E)$ mit der Wurzel w, bestehend aus Knoten $u \in U$ und Kanten $e \in E$, welche die Operationen und Übergänge zwischen Operationen darstellen, Quelle: Eigene Darstellung in Anlehnung an [29].

Der gesamte Baum T_i besteht aus n_R Routen $R_i := \{R_{i1}, R_{i2}, \ldots, R_{ib}, \ldots, R_{in_R}\}$, im Beispiel definiert durch $R_i := \{R_{i1}, R_{i2}, R_{i3}, R_{i4}\}$, wobei sich die unterschiedlichen Routen durch die Operationsflexibilität einiger Operationen ergeben und den Handlungsspielraum des GAs repräsentieren. Jede Route R_{ib} stellt einen gerichteten Weg von der Wurzel w bis zu einem der Blätter des Baumes dar. Ein Blatt bezeichnet hierbei jeweils den letzten Knoten einer Route [232, S. 175]. Die Routen R_i sind hierbei nicht zu verwechseln mit den Prozessplänen P_i. Eine Route eines Auftrags beinhaltet im Vergleich zum Prozessplan zusätzlich die Stationswahl pro Operation. Die Blätter des Baumes T_i repräsentieren in der MP generell die Senke, in welcher fertiggestellte Produkte am Ende abzuliefern sind. Da in der Senke jedoch keine Bearbeitung mehr stattfindet, vernachlässigen die folgenden Abbildungen die Senke aufgrund der einfacheren Darstellung.

Mögliche Produktionsstatus: Je nachdem, in welchem Prozess sich das Produkt aktu-

ell befindet, bleibt ein Teilgraph des in Abbildung 3.5 gezeigten Gesamtgraphen für die Entscheidungen bei der Optimierung übrig. Zum Scheduling-Zeitpunkt t_s gilt für einen Auftrag J_i also: $R_{i,t_s} \subseteq R_i$. In der in Abschnitt 2.4.2 zusammengefassten MP kann ein Produktionsauftrag fünf Status einnehmen. Der jeweilige Status zeigt den Fortschritt des zugehörigen Auftrags. Abbildung 3.6 stellt die fünf Status anhand des oben eingeführten Produktbeispiels dar. Die rote Markierung im Bild kennzeichnet den aktuellen Prozess. Abbildung 3.6a zeigt einen neu eingelasteten Auftrag mit dem Status „Neu“. Der verfügbare Teilgraph T_i' entspricht dem Gesamtgraphen T_i. Abbildung 3.6b zeigt einen Auftrag, der sich gerade in Bearbeitung (Status „IB“) befindet. Vom aktuellen Knoten aus bleiben zwei Routen zur Wahl. Abbildung 3.6c zeigt mehrere Produktionsstatus: „EP“, „AP“ und „FTF“, alle drei repräsentiert von derselben Kante des Graphen, weil sie zum Übergang zwischen zwei Operationen zählen. Liegt ein Produkt also aktuell im EP oder AP einer Station oder befindet sich auf dem Transportweg dazwischen, ergibt sich der Teilgraph T_i' von der aktiven Kante aus zeitlich nach vorne schauend. Auch hier bleiben im Beispiel zwei Routen zur Wahl.

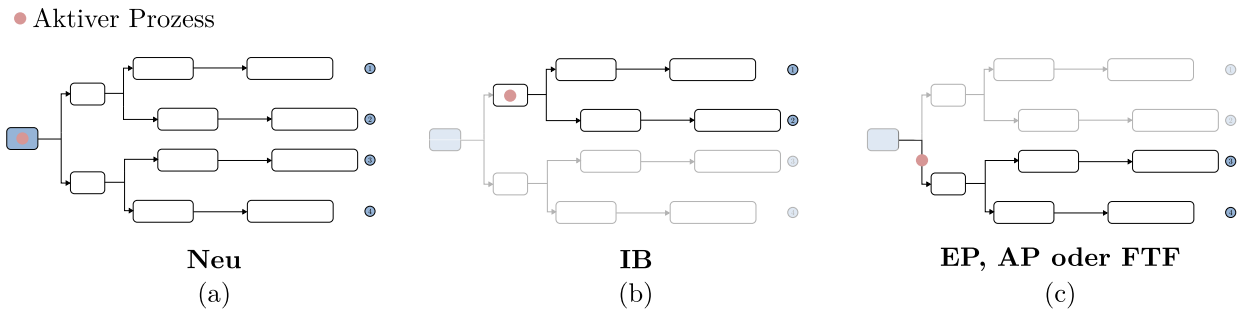


Abbildung 3.6: Fünf mögliche Produktionsstatus „Neu“ (a), „IB“ (b) und „EP“, „AP“ sowie „FTF“ (c), sowie die jeweils verbleibenden Teilgraphen.

Verbleibende Prozesszeiten: Für eine exakte Schedule-Berechnung benötigt die Scheduling-Einheit zu jedem aktiven Auftrag Auskunft über die ausstehende Prozesszeit des aktuellen Prozesses zum Zeitpunkt t_s. Zu den Prozessen zählen hier sowohl Knoten als auch Kanten. Tabelle 3.3 dokumentiert die fünf Status und wie ihre verbleibenden Prozesszeiten $t_{i,\text{verbl}}$ zu berechnen sind. Für den Status „Neu“ beträgt $t_{i,\text{verbl}}$ der Transportzeit von der Quelle zur nächsten Station, einschließlich der Mindest-Liegezeit im EP der anvisierten Station. Dies entspricht dem frühestmöglichen Zeitpunkt des Übergangs in die Bearbeitungsposition der nächsten Station, ohne dabei die Aufträge zu beachten, die sich bereits an der Zielstation aufhalten. Befindet sich ein Auftrag J_i bereits im EP, beträgt $t_{i,\text{verbl}}$ analog entweder der verbleibenden Transportzeit auf dem Fließband des EPs oder, wenn das Produkt nach dem Fließbandtransport schon vor dem Prozessplatz bereitliegt, der verbleibenden Bearbeitungszeit $t_{i'p'l',\text{Rest}}$ des sich auf dem Bearbeitungsplatz befindenden Produkts $J_{i'}$. Hierbei werden Abhängigkeiten innerhalb des Puffers vorerst vernachlässigt. Die Berechnung erfolgt damit identisch für alle sich im selben Puffer befindenden Produkte, wobei die Abhängigkeiten zwi-

schen Produkten in der später vorgestellten Decodierungsfunktion des GAs berücksichtigt werden. Für den Status „IB“ nimmt die Scheduling-Einheit die verbleibende Bearbeitungszeit $t_{ipl,\text{Rest}}$ des sich in Bearbeitung befindenden Auftrags J_i als $t_{i,\text{verbl}}$ an. Die Berechnung für „AP“ und „FTF“ erfolgt analog zum Status „Neu“ bis zum frühestmöglichen Zeitpunkt, zu dem das jeweilige Produkt an der gewählten Station in Bearbeitung gehen könnte. Auch hier werden die Aufträge, die sich bereits an der Zielstation befinden, vorerst nicht betrachtet. Die verbleibenden Prozesszeiten der Status „Neu“, „AP“ und „FTF“ sind damit abhängig von der gewählten Route R_{ib} des Auftrags J_i.

Tabelle 3.3: Mögliche Status eines Produktionsauftrags J_i und die jeweilige Berechnung der verbleibenden Prozesszeit $t_{i,\text{verbl}}$.

Status	**Beschreibung**	**Berechnung** $t_{i,\text{verbl}}$
Neu	Auftrag in Quelle erzeugt	$= t_{\text{TR},Quelle \rightarrow M_k} + t_{\text{EP},min}$
EP	Auftrag in EP einer Station	$= max(t_{\text{EP,Rest}}, t_{i'p'l',\text{Rest}})$
IB	Auftrag in Bearbeitung an Station	$= t_{ipl,\text{Rest}}$
AP	Auftrag in AP einer Station	$= t_{\text{AP,Rest}} + t_{\text{TR},M_{k1} \rightarrow M_{k2}} + t_{\text{EP},min}$
FTF	Auftrag auf dem Transportweg mittels FTF	$= t_{\text{TR},M_{k1} \rightarrow M_{k2},\text{Rest}} + t_{\text{EP},min}$

Alle nicht den aktuellen Prozessen entsprechenden Prozesszeiten des entstehenden Teilgraphen T_i' eines Auftrags J_i, sowohl für Operationen als auch für die Übergänge zwischen Stationen, entsprechen den Minimalwerten des ursprünglichen Produktgraphen aus Abbildung 3.5. So kann jedem Knoten u und jeder Kante e des Graphen eine frühestmögliche Startzeit $t_{u,\text{start}}$ oder $t_{e,\text{start}}$ und die dazugehörige Endzeit $t_{u,\text{ende}}$ oder $t_{e,\text{ende}}$ zugewiesen werden. Die später in Abschnitt 3.3.2 vorgestellte Decodierungsfunktion des GAs nutzt diese Start- und Endzeiten zum Aufbau des Schedules. Außerdem berücksichtigt sie alle gegenseitigen Auftragsabhängigkeiten, um aus den aufbereiteten Teilgraphen einen gültigen Schedule zu bilden.

Vorausschau: Für jeden aktiven Produktionsauftrag $J_i \in J_{\text{akt}}$ schaut die Scheduling-Einheit basierend auf dem ermittelten Teilgraphen T_i' mit den zugewiesenen Start- und Endzeiten um ΔT_{GA} voraus, um die einzuplanenden Prozesse zu bestimmen. Das Zeitfenster der Vorausschau ΔT_{GA} wird im Folgenden als Ermittlungszeitraum bezeichnet. Abbildung 3.7 verdeutlicht das Prinzip der Vorausschau für einen Auftrag J_i, der sich zum Zeitpunkt t_{s1} in Bearbeitung (Status „IB“) mit $t_{i,\text{verbl}} = t_{ipl,\text{Rest}}$ befindet. Hierbei stehen noch zwei Routen R_{i1} und R_{i2} zur Verfügung. Der Ermittlungszeitraum erfasst den Knoten in Route R_{i2} nicht vollständig. Damit die folgende Optimierung mittels GA jedoch valide Schedules berechnen kann, wird der Knoten trotzdem vollständig berücksichtigt. Dies gilt nicht für die teilweise im Ermittlungszeitraum liegende Kante von Route R_{i1}. Die Planung bezieht nicht vollständig erfasste Kanten nicht in die Optimierung mit ein. Der Grund hierfür ist der spätere Fokus

der Optimierung auf die optimale Einplanung der Knoten, welche den Operationen an den Stationen entsprechen. Die Kanten spielen hier eine untergeordnete Rolle. Sie werden zur Ermittlung der frühestmöglichen Startzeiten von Operationen verwendet.

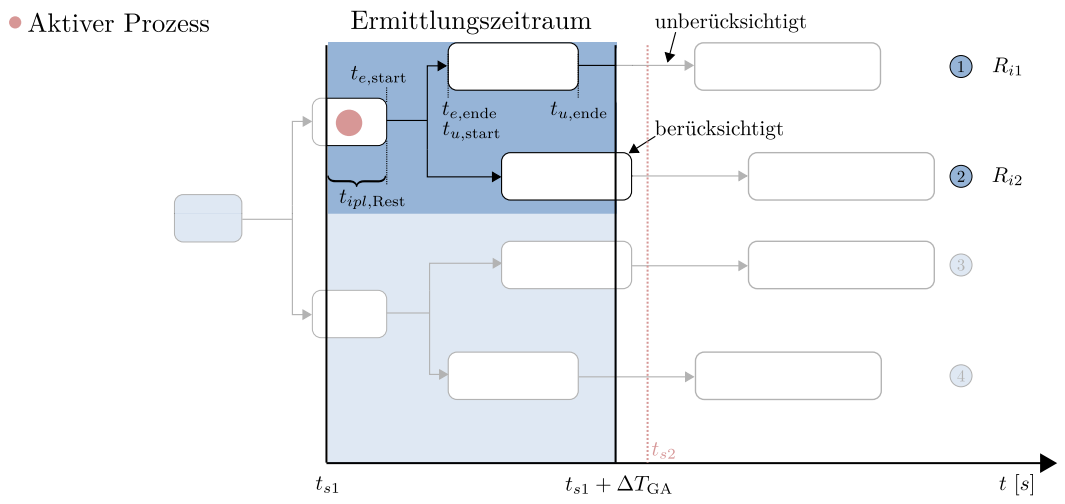


Abbildung 3.7: Darstellung der Datenaufbereitung für einen Auftrag J_i, der sich gerade in Bearbeitung befindet, Quelle: Eigene Darstellung in Anlehnung an [29].

Das Abschneiden nicht vollständig im Ermittlungszeitraum enthaltener Kanten entspricht dem implementierungsbedingten Grund für die Berücksichtigung der maximalen Übergangszeit $UEZ_{ipl,max}$ zur Ermittlung der Mindest-Vorausschau $\Delta T_{\text{GA},min}$ eingangs des Abschnitts in (3.3). Fände die nächste Schedule-Berechnung in Abbildung 3.7 erst im Bereich dieser unberücksichtigten Kante statt, beispielhaft rot dargestellt als Zeitpunkt t_{s2}, würde dem Auftrag bei Fertigstellung des der Kante vorgelagerten Knotens die Entscheidung fehlen, wo der nächste Prozess stattfindet. (3.3) stellt durch die Einplanung von $UEZ_{ipl,max}$ als Worst Case sicher, dass dieser Fall nicht eintritt. Abgeleitet von (3.3) muss also für zwei aufeinanderfolgende Scheduling-Zeitpunkte t_{s1} und t_{s2} gelten:

$$\Delta t_s = t_{s2} - t_{s1} \leq \Delta T_{\text{GA},min} - UEZ_{ipl,max} \quad \forall J_i \in J, \forall P_{ip} \in P_i, \forall O_{ipl} \in O_{ip}. \tag{3.4}$$

Für die folgende Optimierung findet der oben skizzierte Aufbereitungsschritt für jeden aktiven Auftrag statt. Die im Ermittlungszeitraum enthaltenen Routen inkl. der zu planenden Knoten bzw. Operationen bestimmen den Planungsspielraum des GAs.

Optimierung mithilfe des Genetischen Algorithmus

Nachdem die Daten aller aktiven Produktionsaufträge J_{akt} in Graphenform aufbereitet wurden, kann der GA mit der Optimierung beginnen. Abschnitt 3.2 führt die hierfür notwendigen Grundlagen ein. Zuerst stellt sich die Frage nach der Codierung einer Lösung des Optimierungsproblems und der damit einhergehenden Darstellung eines Individuums. Hierfür orien-

tiert sich die Arbeit an den erfolgreichen GA-Implementierungen für Scheduling-Probleme in [230], [231], [235], die zwar nicht das identische Scheduling-Problem dieser Arbeit lösen, jedoch Inspiration dafür bieten, beispielsweise für die Parameterwahl des GAs. Abbildung 3.8 stellt die gewählte Repräsentation dar. Die Länge des Individuums zum Scheduling-Zeitpunkt t_s ist von der Anzahl aktiver Aufträge J_{akt} abhängig. Bei jedem Scheduling-Lauf werden die Individuen somit abhängig von der Größe der Menge J_{akt} initialisiert. Jedes Gen des Individuums repräsentiert einen der aktiven Aufträge. Das Allel, also der Wert jedes Gens, bezeichnet die vom GA gewählte Route R_{ib} aus der durch den Teilgraphen T_i' abgeleiteten Menge R_{i,t_s} verfügbarer Routen für Auftrag J_i im Ermittlungszeitraum ΔT_{GA} (vgl. Abbildung 3.7). Hierbei bezeichnet das Allel den von null beginnenden Index in der Menge R_{i,t_s}. Deshalb ist der Wert des ersten Gens in Abbildung 3.8 drei und nicht vier, obwohl es sich um die vierte verfügbare Route handelt. Für jeden aktiven Auftrag J_i hängt der zulässige Wertebereich W_i des Gens damit von den verfügbaren Routen R_{i,t_s} zum Zeitpunkt t_s ab. Durch die Wahl der Route pro Auftrag für den Ermittlungszeitraum untersucht der GA den gegebenen Lösungsraum. Der GA adressiert damit die Entscheidungskategorie Auftragsrouting auf direktem Wege.

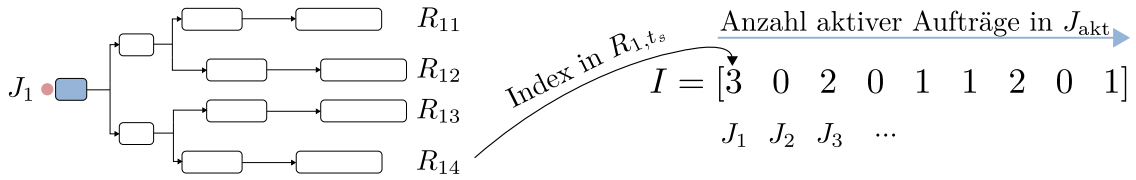


Abbildung 3.8: Codierung eines Individuums I.

Der folgende Ablauf der Optimierung mithilfe des GAs folgt Alg. 3.1. Zuerst ist eine initiale Population zu generieren. Anschließend werden die Individuen decodiert und bewertet. Es folgen Selektion, Rekombination und Mutation, bevor die Population durch die erzeugte Kindpopulation ersetzt wird. Dann beginnt der Zyklus von Neuem, beginnend mit der Evaluierung der erzeugten Kindpopulation. Nach Durchlauf einer festgelegten Anzahl von Generationen N_{GEN} bestimmt das fitteste Individuum den Produktionsschedule. Alternativ kann ein Zeitlimit t_{opt} gesetzt werden, nachdem der GA die Optimierung abbricht und das bis dato fitteste Individuum an die Produktion übergibt. Dies dient zur Einhaltung der Echtzeitbedingung.

Initiale Population: Der Startpunkt des GAs, die Generierung der initialen Ausgangspopulation λ_0 der Individuen mit der Größe N_{POP}, erfolgt zufällig. Dies ermöglicht heterogene Ausgangslösungen. Um ausschließlich zulässige Individuen zu erzeugen, wählt der GA für jeden Auftrag J_i und damit jedes Gen eines Individuums einen zufälligen Wert des zulässigen Wertebereichs W_i aus. Für das dargestellte Beispiel in Abbildung 3.8 liegt dieser Wertebereich beispielsweise für J_1 bei $W_1 = \{0, 1, 2, 3\}$.

Decodierung und Evaluation: Die Bewertung eines Individuums I erfolgt durch Deco-

dierung eines wie in Abbildung 3.8 codierten Individuums und die anschließende Berechnung seiner Fitness $J(I)$ mithilfe einer Fitnessfunktion. Die Decodierung folgt dem Pseudocode in Alg. 3.2 und arbeitet ereignisorientiert. Das vollständig decodierte Individuum repräsentiert eine Lösung des Scheduling-Problems zum Zeitpunkt t_s. Als Basis für die zu planenden Ereignisse dienen die in Abbildung 3.7 aufbereiteten Start- und Endzeiten der Knoten und Kanten der im Individuum I gewählten Routen R_{ib} der aktiven Aufträge J_{akt}. Die Decodierung konzentriert sich hierbei auf die Knoten der Graphen und somit die Operationen der Aufträge. Diese Knoten werden im Folgenden als Schedule-Items bezeichnet. Bei der Decodierung kommen zusätzlich zum Auftragsrouting die Entscheidungskategorien Stationsmanagement und Flottenmanagement hinzu. Sie werden durch die ereignisorientierte Vorgehensweise entschieden. Die Ereignisorientierung bietet sich aufgrund der vielen Abhängigkeiten zwischen aktiven Aufträgen an. Prädizierte Zeitstempel verwalten die Abhängigkeiten und sorgen für zeitliche Korrektheit des Schedules.

Algorithmus 3.2 Decodierung eines Individuums in Anlehnung an [30].

1: **Input**: Individuum I mit gewählten Routen R_{ib} aller aktiven Aufträge J_{akt}.
2: Platziere erste Schedule-Items der Aufträge mit Status „IB“.
3: Platziere erste Schedule-Items der Aufträge mit Status „EP“ unter Berücksichtigung der FIFO-Reihenfolge.
4: Aktualisiere Zeitstempel der verbleibenden Items der Aufträge J_i im Status „EP“.
5: Sammle verbleibende Schedule-Items $Items(I)$ aller aktiven Aufträge J_{akt}.
6: Sortiere die Liste $Items(I)$ nach den Zeitstempeln der Schedule-Items.
7: **while** $Items(I)$ ist nicht $\emptyset$ **do**
8: Platziere erstes Item der sortierten Liste im Schedule.
9: Entferne das platzierte Item aus der Liste $Items(I)$.
10: Aktualisiere Zeitstempel der Folgeitems des betroffenen Auftrags J_i.
11: Sortiere die Liste $Items(I)$ nach den Zeitstempeln der Schedule-Items.
12: **end while**

Die Abläufe der Decodierungsfunktion aus Alg. 3.2 sind definiert zu:

- Zuerst setzt die Decodierungsfunktion die Schedule-Items, die bereits vom aktuellen Produktionsstatus zum Zeitpunkt t_s vordefiniert sind. Erstens zählen hierzu die ersten Items der sich in Bearbeitung befindenden Aufträge, somit der jeweilige aktive Knoten im zugehörigen Graphen. Zweitens werden alle Schedule-Items der Aufträge gesetzt, die sich in einem EP befinden, da sich deren Reihenfolge aufgrund des FIFO-Prinzips des Puffers nicht mehr ändern kann. In diesem Fall wird das zu setzende Item durch den Folgeknoten der aktiven Kante im zugehörigen Graphen repräsentiert. Die Zeitstempel der Folgeitems der Aufträge in EPs müssen nach diesem Schritt in Abhängigkeit von der zugewiesenen Startzeit des gesetzten Items aktualisiert werden. Die Aktualisierung aller Start- und Endzeiten des betroffenen Auftrags J_i basiert auf der Differenz der geplanten Startzeit $t_{u,\text{start}}$ des gesetzten Items und der tatsächlich zugewiesenen Start-

zeit $t'_{u,\text{start}}$. Alle Folgeitems des betroffenen Auftrags J_i werden dann um $t'_{u,\text{start}} - t_{u,\text{start}}$ zeitlich nach hinten versetzt. Diese Aktualisierung ist notwendig, damit für die Folgeitems von J_i eine realistische und aktualisierte Schätzung der frühestmöglichen Start- und Endzeiten besteht. Die Unterschiede in den Startzeiten entstehen durch die in Abbildung 3.7 nicht berücksichtigten Abhängigkeiten zwischen den Aufträgen, welche an dieser Stelle relevant werden. Deshalb ist auch im weiteren Verlauf zu beachten, dass die in Abbildung 3.7 erarbeiteten Start- und Endzeiten der Folgeitems einer gewählten Route R_{ib} eines Auftrags J_i bei jedem gesetzten Item bzw. Knoten u um $t'_{u,\text{start}} - t_{u,\text{start}}$ zu aktualisieren sind.

- Nach Setzen der ersten Items der Aufträge mit Status „IB“ und „EP“ sammelt die Decodierungsfunktion alle Schedule-Items $Items(I)$, die noch nicht im Schedule platziert wurden. Die aktuell geschätzten Start- und Endzeiten der Items werden dann dazu verwendet, die Liste der zu setzenden Schedule-Items zeitlich chronologisch zu sortieren[2]. Diese zeitliche Sortierung sorgt dafür, dass die Operationen der pro Auftrag gewählten Route nach dem FCFS-Prinzip an den Stationen der MP eingeplant werden. Hierdurch entscheidet sich das Stations- und Flottenmanagement in Form der geplanten Operationsreihenfolgen und Transporte.

- Ab diesem Punkt wiederholen sich die Vorgänge, bis $Items(I)$ keine Schedule-Items mehr enthält mit den folgenden Abläufen: Die Decodierungsfunktion platziert das erste anstehende Schedule-Item im Schedule und löscht es aus der Liste. Außerdem erfolgt eine Aktualisierung der Folgeitems des gesetzten Items im betroffenen Auftrag J_i. Es folgt eine Aktualisierung der in $Items(I)$ verbliebenen Items nach ihren Zeitstempeln und die Schleife wiederholt sich rekursiv.

Die Variation der Scheduling-Lösungen in Form der Individuen erfolgt damit ausschließlich auf Basis der gewählten Routen. Stationsmanagement und Flottenmanagement entscheiden sich indirekt mithilfe des angewendeten FCFS-Prinzips. Diese Vereinfachung macht erstens Sinn, weil in den EPs der Stationen das FIFO-Prinzip gilt und damit der Spielraum der Scheduling-Lösungen im Bereich der Reihenfolge der Abarbeitung von Aufträgen eingeschränkt ist. Aufträge, die sich bereits in einem EP befinden, können problemstellungsbedingt nicht mehr variiert werden. Zweitens gilt es, gesamtheitlich den Durchsatz zu maximieren und damit nach dem Gesetz von Little (vgl. (2.5)) die Durchlaufzeiten zu minimieren. Scheduling-Lösungen, die nicht nach dem FCFS-Prinzip arbeiten, ermöglichen hingegen, dass beispielsweise ein früher ankommender Auftrag warten muss, damit ein später eintreffender Vorrang erhält. Damit erhöht sich die Durchlaufzeit des wartenden Auftrags, obwohl er schneller sein könnte. Da alle Aufträge identisch priorisiert werden und es beispielsweise keine Eilaufträge gibt, ist dies kontraproduktiv hinsichtlich der gesamtheitlichen Durchsatzoptimierung.

[2]Beim unwahrscheinlichen Eintreten identischer Zeitstempel behält die Implementierung für die betroffenen Elemente die aktuelle Reihenfolge in $Items(I)$ bei.

Einen beispielhaften decodierten Schedule zeigt Abbildung 3.9 für fünf Aufträge von drei Produkttypen, fünf FTFs und zwei Stationen. Auch AP und EP der Stationen sind abgebildet. Sie sind zu berücksichtigen, weil sie wie die Transporte zur Übergangszeit zählen und einer Kante im jeweiligen Graphen zuzuordnen sind. Die Nummerierung der Items repräsentiert die über die Zeitstempel festgelegte Reihenfolge der gesetzten Items bei der Decodierung.

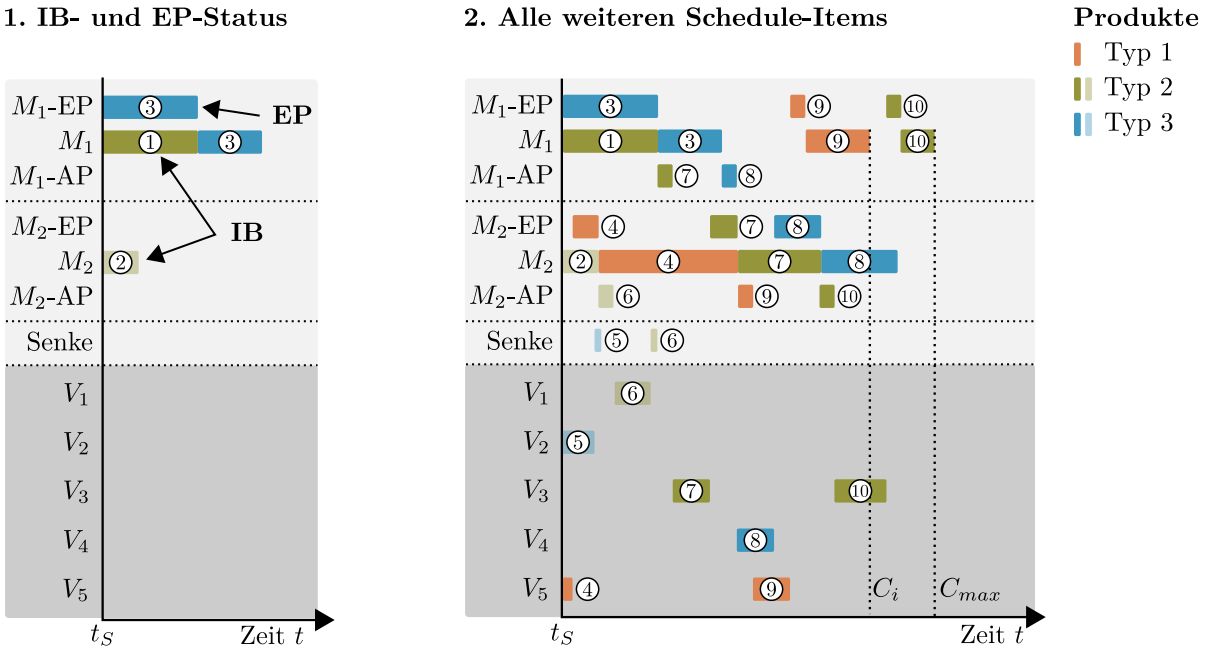


Abbildung 3.9: Decodierung eines Individuums für fünf Produktionsaufträge bestehend aus drei Produkttypen mit den zuerst gesetzten Schedule-Items auf der linken Seite und allen weiteren hinzugefügten Items rechts.

Die linke Seite zeigt den ersten Schritt der Decodierung. Hier setzt Alg. 3.2 die ersten Schedule-Items der Aufträge, die sich entweder in Bearbeitung (Items 1 und 2) oder im EP (Item 3) befinden. Beim Status „EP“ setzt die Decodierungsfunktion hierbei sowohl die Liegezeit im Puffer M_1-EP als auch die folgende Bearbeitung an M_1. Außerdem ist zu sehen, dass die Startzeit von Item 3 an M_1 abhängig von der Bearbeitung von Item 1 nach hinten versetzt wird, was erstens die Liegezeit im EP verlängert und sich zweitens auf die Folgeitems des blauen Auftrags auswirkt, deren Zeitstempel zu aktualisieren sind. Der rechte Teil der Abbildung zeigt dann den vollständig decodierten Schedule nach Alg. 3.2 (zusätzliche Items 4 bis 10). Zu jedem Schedule-Item setzt die Decodierungsfunktion hier auch den vorgeschalteten Stationsübergang in Form des zugehörigen Transports auf dem zugewiesenen Fahrzeug sowie der verbrachten Zeiten im EP und AP. Damit entsteht ein gesamtheitliches Bild der MP. Der hellblaue Auftrag befindet sich beispielsweise auf FTF V_2 und wird zur Senke transportiert (Item 5). Der orange-farbene Auftrag befindet sich hingegen im Transport zu M_2 und wartet dann im EP, bis er in Bearbeitung gehen kann (Item 4).

Die Bewertung der decodierten Individuen I erfolgt mithilfe folgender Fitnessfunktion:

$$J(I) = -(\beta \cdot C_{max} + \frac{\sum_{i=1}^{n_J} C_i}{n_J}). \tag{3.5}$$

C_{max} bezeichnet die Endzeit der spätesten Operation im Schedule, vgl. Abbildung 3.9. C_i bezeichnet hingegen die Endzeit der spätesten Operation eines spezifischen Auftrags J_i der n_J aktiven Aufträge J_{akt}, im Bild beispielhaft dargestellt für den orange-farbenen Auftrag. Sowohl C_{max} als auch C_i entsprechen hierbei der zeitlichen Differenz zum aktuellen Scheduling-Zeitpunkt t_s. β gewichtet die beiden Terme. Der erste Term der Gleichung repräsentiert die gesamte Produktionsspanne des Schedules, wobei der zweite Term die durchschnittliche auftragsindividuelle Produktionsspanne darstellt. Damit entsprechen beide Terme der gleichen Größenordnung. Für die vorliegende Arbeit hat sich eine Gewichtung mit $\beta = 1$ als empirisch zuverlässig gute und robuste Lösung herauskristallisiert. Insgesamt stellt die Kombination der beiden Terme sicher, dass nicht nur die gesamte Produktionsspanne, sondern auch die individuellen Durchlaufzeiten optimiert werden, um die Durchsatzoptimierung nach dem Gesetz von Little zu erzielen.

Selektion, Rekombination und Mutation: Nach der Erläuterung der Codierung, der Decodierung und der Bewertung der Individuen folgen die GA-Mechanismen der Selektion, Rekombination und Mutation zur Untersuchung des Suchraums des Optimierungsproblems. Für eine gute GA-Performanz sind diese Mechanismen i.d.R. auf das spezifische Problem anzupassen. Für die in Abbildung 3.8 geschickt gewählte Codierung trifft dies allerdings nicht zu. Sie ermöglicht die Anwendung der in Abschnitt 3.2 eingeführten Standard-Operatoren, weil für jedes Individuum eines Optimierungslaufes die Reihenfolge der aktiven Aufträge J_{akt} im Individuum identisch bleibt. Ein einzelnes Gen repräsentiert in jedem Individuum denselben aktiven Produktionsauftrag. Die Wahl der Standard-Operatoren zeigt ein gutes Konvergenzverhalten, wie die späteren Untersuchungen dokumentieren. Der GA arbeitet dabei mit folgenden Mechanismen:

- **Selektion**: Pezzella et al. [235], Zhang et al. [236] und Shao et al. [231] schlussfolgern die Wettkampfselektion als beste Selektionsmethode in ihren Scheduling-Anwendungen. Dies zeigte sich auch im vorliegenden Scheduling-Problem. Aus der Menge an N_{POP} Individuen werden hierzu $N_{\text{SEL}} = N_{\text{POP}}$ Wettkämpfe mit Turniergröße zwei durchgeführt. In jedem Wettkampf gewinnt das fittere Individuum.

- **Rekombination**: Mit einer Wahrscheinlichkeit von p_R rekombinieren sich zwei Individuen mithilfe des in Abschnitt 3.2 vorgestellten 2-Punkt-Crossovers, wobei die Auswahl der Crossover-Punkte zufällig erfolgt.

- **Mutation**: Mit einer Wahrscheinlichkeit von p_M durchläuft ein Individuum eine an die in Abbildung 3.2 angelehnte Mutation. Dabei wählt der GA für ein zufällig ausgewähl-

tes Gen des Individuums eine neue zufällige Route R_{ib} der zum aktuellen Zeitpunkt t_s verfügbaren Routen R_{i,t_s}.

Da der GA nur den zulässigen Bereich verfügbarer Routen betrachtet, behält jedes Individuum seine Gültigkeit. Nach Durchlauf der drei Mechanismen ersetzt die erzeugte Kindpopulation die Elternpopulation. Diese Abläufe werden N_{GEN} Mal wiederholt oder alternativ bei einem Zeitlimit von t_{opt} abgebrochen.

Optimierter Schedule: Am Ende wird das Individuum mit der höchsten Fitness ausgewählt. Damit gewinnt das Individuum I, welches (3.2) unter Berücksichtigung der Fitnessfunktion aus (3.5) maximiert:

$$J_{max} = \max_{I \in D} -(C_{max} + \frac{\sum_{i=1}^{n_J} C_i}{n_J}), \tag{3.6}$$

wobei in (3.5) $\beta = 1$ für die vorliegende Arbeit gilt. Im nächsten Schritt sind die vom GA optimierten Entscheidungen an die Produktion zu übergeben.

Übergabe der getroffenen Entscheidungen

Das mittels GA erzielte Optimierungsergebnis gibt die zum Zeitpunkt t_s in der MP anstehenden Entscheidungen vor. Hierzu wird das Individuum mit der besten Fitness decodiert und in den Schedule überführt. Aus dem optimierten Schedule lassen sich die Routenwahl jedes Auftrags inklusive der Transporte auf den Fahrzeugen sowie die Sequenzierung und Terminierung der Operationen an den Stationen ableiten. Damit enthalten die Entscheidungen Informationen zu den Kategorien Stationsmanagement, Auftragsrouting und Flottenmanagement. Diese Entscheidungen gehen in die Produktion über.

Die Schedule-Optimierung mithilfe des GAs als Teil des rollierenden Scheduling-Konzepts lässt sich folgendermaßen zusammenfassen: Zu einem Scheduling-Zeitpunkt t_s liest der GA den aktuellen Status aller aktiven Produktionsaufträge J_{akt} aus, schaut für jeden dieser Aufträge um ΔT_{GA} voraus und berechnet den Teilgraphen jedes Auftrags für diesen Bereich. Aus den zur Verfügung stehenden Routen R_{i,t_s} der Aufträge J_i bildet der GA anschließend Generation für Generation bessere Routenkombinationen, um einen möglichst durchsatzoptimalen Schedule zu erzielen. Das Ergebnis der Optimierung wird der Produktion anschließend zur Ausführung übergeben. Einen beispielhaften Schedule dokumentiert Abbildung 3.9. Der nächste Abschnitt schließt den zentralen Ansatz mit der Integration von Störungen ab.

3.3.3 Störungsmanagement

Der rollierende Ansatz arbeitet nicht nur zeitlich periodisch, sondern berechnet Schedules auch bei den auftretenden Ereignissen der Auftragseinlastung sowie bei Störungsstart

$t_{\text{stoer,start}}$ und bei Störungsende $t_{\text{stoer,ende}}$ einer Station. Die störungsbedingten Schedule-Berechnungen sind damit dem Störungsmanagement zuzuordnen.

Bei beiden Ereignissen berechnet die Steuerung einen neuen Schedule mithilfe des vorgestellten GAs. Der GA berücksichtigt die Störungen hierzu im Rahmen der Vorverarbeitung des Produktionsstatus. Die aufgrund der Störungen angewendeten Adaptionen in der Vorverarbeitung im Vergleich zu Abschnitt 3.3.2 reduzieren sich zu:

- Für alle Produktionsaufträge, die sich im Status „IB“ an einer gestörten Station befinden, platziert der GA lediglich die aktuell gestörte Operation im Schedule. Als verbleibende Prozesszeit $t_{i,\text{verbl}}$ verwendet der GA die theoretisch verbleibende Bearbeitungszeit $t_{ipl,\text{Rest}}$, ohne Rücksicht auf die Unterbrechung durch die Störung zu nehmen. Waren vor der Störung also noch $20s$ Bearbeitung ausstehend, ist $t_{i,\text{verbl}} = 20s$. Die weiter in der Zukunft liegenden Produktionsschritte finden keine Berücksichtigung.

- Für alle Produktionsaufträge, die sich im EP einer gestörten Station, auf einem FTF zu einer gestörten Station oder im AP[3] der Vorgängerstation mit dem Ziel der gestörten Station befinden, platziert der GA ebenfalls lediglich den Folgeknoten der aktiven Kante und damit die geplante Operation an der gestörten Station im Schedule. Auch hier findet keine Vorausschau in die Zukunft der betroffenen Aufträge statt.

Diese beiden Anpassungen sorgen dafür, dass die von den Störungen blockierten Aufträge nicht in die Zukunft geplant werden, sondern nach der Operation an der gestörten Station vorübergehend enden. Dadurch steht die Produktionskapazität, welche sonst von den abgeschnittenen Zukunftsoperationen teilweise blockiert wäre, stattdessen anderen Aufträgen zur Verfügung. In die Fitnessfunktion des GAs gehen die unmittelbar von einer Störung betroffenen Aufträge J_i durch ihre jeweils individuelle Produktionsspanne C_i ein. Für diesen Fall ist C_i anhand des Bearbeitungsendes der jeweils einen an der gestörten Station eingeplanten Operation zu bestimmen. Abgesehen von den genannten Adaptionen führt die zentrale Scheduling-Einheit die Vorverarbeitung aus Abschnitt 3.3.2 inkl. der Berücksichtigung des FIFO-Prinzips in den Puffern für die blockierten Aufträge durch.

Für alle anderen Aufträge, die obige Bedingungen nicht erfüllen und von der Störung nicht unmittelbar betroffen sind, erfolgt die Planung vollständig, wie in Abschnitt 3.3.2 beschrieben. Der GA führt hierbei aus drei Gründen nicht alle Aufträge an den gestörten Stationen vorbei: Erstens bewirkt die Einplanung der jeweils direkt von der Störung betroffenen Operationen (s. Bedingungen oben), dass der GA anhand seiner Fitnessfunktion automatisch beginnt, die Station zu vermeiden, wenn dort die eingeplante Kapazität steigt. Je mehr Operationen direkt von der Störung betroffen sind, desto stärker wird die Station vermieden.

[3] Im AP könnte theoretisch noch ein Umrouting stattfinden. Jedoch befindet sich der betroffene Auftrag implementierungsseitig schon auf der Kante zur nächsten gewählten Station. Damit ist die Entscheidung für die nächste Station bereits getroffen. Da der Status „AP“ aufgrund der relativ kurzen Puffer-Transportzeit selten auftritt, beeinträchtigt dies nicht die Lösungsgüte.

Die bereits nach obigen Bedingungen geplanten Operationen führen hierbei dazu, dass bei Einplanung zusätzlicher Operationen die Belastung der Station steigt und die Durchlaufzeiten dieser zusätzlich eingeplanten Aufträge ebenfalls ansteigen. Da die Fitnessfunktion aus (3.5) Durchlaufzeiten minimiert, um den Durchsatz zu maximieren, verhindert sie, dass zu viele Aufträge trotz der Störung an die Station geschickt werden. Dadurch nutzt der GA die gestörte Station und die Alternativstationen möglichst gleichmäßig. Somit berücksichtigt er auch implizit, dass eine Alternativstation nicht beliebig viele Aufträge prozessieren kann. Denn es ist unwahrscheinlich, dass eine Alternativstation alle Aufträge, die ansonsten auf zwei Stationen bearbeitet werden, während der Störungszeit einer dieser Stationen abarbeiten kann. Zweitens können auch Stationen existieren, für die keine Alternativen existieren. Geht eine derartige Station in Störung, muss trotzdem auf dieser Station geplant werden. Drittens ist ein Störungsende nicht vorhersehbar. Dies führt zu Szenarien, in denen eine Fahrt zur gestörten Station förderlich für die Durchsatzoptimierung ist, weil die dortige Störung möglicherweise bald zu Ende ist. In diesem Fall kann es also Sinn ergeben, an einer gestörten Station bereits einen zu bearbeitenden Auftrag bereitstehen zu haben, um einen Leerlauf an der Station zu vermeiden. Im Vergleich zu alternativen Lösungen, in denen beispielsweise eine gestörte Station vollständig vermieden wird, zeigte sich das hier vorgestellte Störungsmanagement als empirisch verlässlichste Lösung.

Abschließend fasst Abbildung 3.10 die Zusammenhänge schematisch zusammen. Zur einfacheren Darstellung blendet die Abbildung die Fahrzeuge hierfür aus. Unter der Annahme

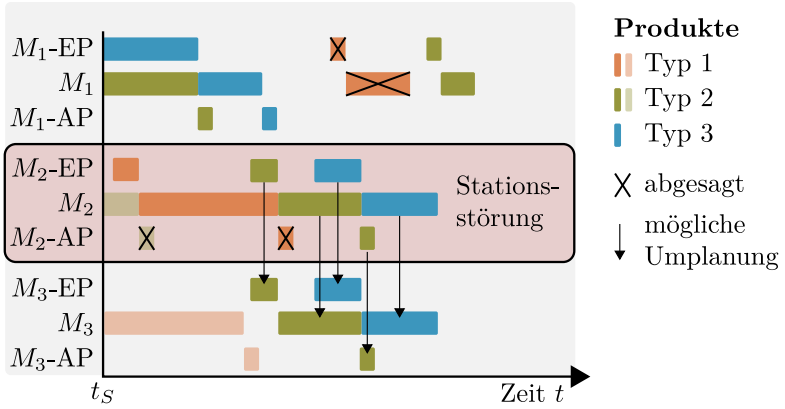


Abbildung 3.10: Darstellung einer Störsituation an Station M_2, wobei mit M_3 eine Alternativstation zur Verfügung steht.

einer Störung an Station M_2 und einer verfügbaren Alternativstation M_3 ergeben sich nach obigem Prinzip folgende Anpassungen im Schedule: Alle durchgekreuzten Elemente betrachtet der GA nicht, weil die entsprechenden Aufträge direkt von der Störung betroffen sind und deshalb bis auf die gestörte Operation selbst nicht weiter in die Zukunft zu planen sind. Die Zukunftsoperationen müssen aus dem Schedule entfernt werden. Davon betroffen sind der hellbraune und der dunkelorange-farbene Auftrag: Der hellbraune Auftrag ist an

der gestörten Station M_2 bereits in Bearbeitung, der dunkelorange-farbene ist schon auf dem Weg dorthin (nicht eingeblendetes FTF). Nach den oben eingeführten Bedingungen plant der GA deshalb die jeweils an der gestörten Station auftretenden Operationen ein, alle Zukunftsoperationen allerdings nicht. Hingegen kann sich für den dunkelbraunen und den dunkelblauen Auftrag, welche ebenfalls an der gestörten Station eingeplant sind, die Route noch ändern. Damit könnte sich der GA dazu entscheiden, sie an der Alternativstation M_3 einzuplanen, dargestellt über die Pfeilverbindungen. Dieses Umrouting führt der GA lediglich durch, wenn sich dadurch die Fitnessfunktion verbessert. Zu beachten sind hier auch möglicherweise unterschiedliche Transportzeiten zu M_2 und M_3.

3.4 Zusammenfassung und Einordnung des Ansatzes

Abschnitt 3.1 legt den grundlegenden Stand der Forschung zentraler Steuerungsansätze dar, auf dessen Grundlage in Abschnitt 3.1.3 ein Grobkonzept abgeleitet wird. Anschließend beschreibt Abschnitt 3.2 die Grundlagen zu GAs als gewählte Methode für die zentrale Schedule-Berechnung. Abschnitt 3.3 nutzt diese Methodik zur Beschreibung des Feinkonzepts der zentral rollierenden Steuerung, wobei zeitlich periodisch sowie bei Produktionsereignissen wie Stationsstörungen eine Neuberechnung des Schedules stattfindet. Der berechnete Schedule betrachtet hierbei einen zu definierenden Zeithorizont von ΔT_{GA}. Die folgenden Stichpunkte ordnen den zentralen Ansatz in die in Abschnitt 2.4.2 gestellten Anforderungen an eine MP-Steuerung ein. Die einzelnen Punkte beschreiben hierbei entweder, wie die bereits beschriebenen Inhalte die Anforderungen adressieren, oder wie sie in der späteren Implementierung in Abschnitt 6.2.2 umzusetzen sind:

- **Auftragseinlastung**: Die Auftragseinlastung orientiert sich am Conwip-Prinzip. Abschnitt 3.3.1 beschreibt die konzeptionellen Zusammenhänge. Neben der Festlegung der Einlastungszeitpunkte weist die zentrale Einlastungseinheit einem einzulastenden Produkt ein FTF und einen WT zu, welche das Produkt für sämtliche Produktionsoperationen begleiten.

- **Stationsmanagement, Auftragsrouting und Flottenmanagement**: Die an die Einlastung anschließenden Entscheidungen zu Stationsmanagement, Auftragsrouting und Flottenmanagement sind Ergebnis der vom GA rollierend berechneten Schedules. Die Codierung sowie Decodierung der Individuen repräsentiert hierbei die Entscheidungsflexibilität. Die Inhalte hierzu finden sich in Abschnitt 3.3.2.

- **Störungsmanagement**: Das Störungsmanagement erfolgt ebenfalls durch angestoßene Neuberechnungen, wenn eine Störung entweder auftritt oder zu Ende geht, beschrieben in Abschnitt 3.3.3.

- **Durchsatzmaximierung**: Die Durchsatzmaximierung wird durch die Fitnessfunktion des GAs in Abschnitt 3.3.2 erreicht. Sie ist auf eine Durchlaufzeitminimierung ausgelegt, was wiederum nach dem Gesetz von Little zum Durchsatzmaximum führt.

- **Echtzeitfähigkeit**: Echtzeitfähigkeit ermöglicht grundsätzlich die Kombination aus GA als Metaheuristik zur Suche von guten Näherungslösungen und dem Ermittlungszeitraum ΔT_{GA} zur Reduzierung der Problemkomplexität durch zeitliche Dekomposition des Problems. Zur Sicherstellung der Echtzeitfähigkeit erfolgt eine Auslegung der GA-Parameter in Abschnitt 6.3.1 anhand von Messungen der Rechenzeiten beim Scheduling. Hierbei hängen die GA-Parameter von der Problembeschaffenheit ab. Sie sind somit auf ein zu lösendes Steuerungsproblem zu adaptieren.

- **Datenverfügbarkeit**: Datenverfügbarkeit zeigt sich später in der beschriebenen Implementierung in Abschnitt 6.2.2. Für den zentralen Steuerungsansatz ist hierfür die zentrale Datenablage des aktuellen Produktionsstatus notwendig, damit die Datenaufbereitung aus Abschnitt 3.3.2 für die Optimierung möglich ist.

- **Generalisierbarkeit**: Generalisierbarkeit auf unterschiedliche Produktionskonfigurationen im Rahmen der in Abschnitt 2.4.2 definierten Systemgrenze zeigt später Abschnitt 6.5. Folgende Punkte des Steuerungskonzepts ermöglichen derartige Generalisierungsexperimente: Einerseits ist die Auftragseinlastung flexibel, weil sie sich auf einen gegebenen Produktmix einstellt. Andererseits ermöglicht die Flexibilität der GA-Codierung, welche beliebige Produktionsaufträge im Rahmen der Systemgrenze verarbeiten kann, den Umgang mit unterschiedlichen Produkttypen im gegebenen MP-Umfeld. Auch Variationen in der Anzahl der vorhandenen Stationen sowie in der Verteilung der in der MP vorhandenen Stationsfähigkeiten berücksichtigt der GA durch die berechneten Produktgraphen, welche wiederum Einzug in die GA-Codierung finden.

Offen bleiben die Fragen zur Parametereinstellung und Implementierung. Die einzustellenden Parameter legt Abschnitt 6.3.1 fest. Hierzu zählen GA-Parameter wie die Populationsgröße N_{POP} sowie Parameter des rollierenden Prinzips wie der Ermittlungszeitraum ΔT_{GA} und der Abstand zwischen zwei Schedule-Berechnungen Δt_s. Das Implementierungskonzept als zweite offene Frage ist Thema in Abschnitt 6.2.2. Dieser Abschnitt beschreibt eine realitätsnahe Implementierung, die die Erfüllung der Anforderungen an eine realitätsnahe MP-Steuerung sicherstellt.

Der entwickelte zentrale Ansatz deckt die erste der drei Steuerungsarchitekturen ab, welche die vorliegende Arbeit betrachtet. Das nächste Kapitel konzipiert einen dezentralen Ansatz.

4 Dezentraler Ansatz: Auktionsbasiertes Multiagentensystem

Dieses Kapitel erarbeitet einen dezentralen Ansatz zur Steuerung der MP in Form eines auktionsbasierten MAS und leistet damit einen Beitrag zur zweiten Forschungsfrage aus Abschnitt 1.2. Zuerst betrachtet Abschnitt 4.1 den bestehenden Stand der Forschung dezentraler Steuerungsansätze und schlussfolgert ein dezentral organisiertes Grobkonzept. Anschließend erläutert Abschnitt 4.2 die verwendete Kernmethode zur Entscheidungsfindung im dezentralen Steuerungskonzept. Abschnitt 4.3 beschreibt im Folgenden das entwickelte Feinkonzept des auktionsbasierten MAS, bevor Abschnitt 4.4 den Ansatz zusammenfasst und mit den Anforderungen aus Abschnitt 2.4.2 abgleicht.

4.1 Stand der Forschung dezentraler Steuerungsansätze

Dieser Abschnitt fasst den Stand der Forschung für dezentrale Steuerungsansätze in Bezug auf die MP zusammen. Abschnitt 4.1.1 verschafft hierzu einen Überblick zu den in der Literatur verwendeten Lösungsmethoden. Abschnitt 4.1.2 analysiert anschließend ausgewählte dezentrale Steuerungsansätze. Abschließend fasst Abschnitt 4.1.3 die untersuchten Ansätze zusammen, ordnet sie in die Arbeit ein und zieht Schlussfolgerungen für den in Abschnitt 4.3 entwickelten dezentralen Steuerungsansatz.

4.1.1 Überblick zu dezentralen Steuerungsansätzen

Abschnitt 2.2.3 führt reaktive, prädiktiv-reaktive, robust-proaktive, rollierende und MAS-Ansätze für dynamisches Scheduling ein. Letzteres Lösungsfeld entwickelte sich in den letzten Jahren, vor allem im Kontext von Industrie 4.0 und CPPS, zu einer dominierenden Methode im Scheduling [75], [101], [237]. MAS-Ansätze basieren auf einer dezentralen Organisation mehrerer kollaborierender Agenten. MAS sind somit die Kerntechnologie dezentraler Steuerungsansätze [88].

Die Lösungsfindung in MAS kann im dynamischen Scheduling durch verschiedene Arten der Kollaboration und Organisation der Agenten erfolgen. Für Rossit et al. [101] entwickelt sich die dezentrale Entscheidungsfindung im Multiagenten-Scheduling hin zu drei grundlegenden Methoden: spieltheoretisch organisierte Agentensysteme, wobei meist Auktionen zur Verhandlung zwischen Agenten eingesetzt werden, lernende Agenten, die das Steuern der Produktionsanlage erlernen, und Agentenarchitekturen als Rahmenwerke für die dezentrale Entscheidungsfindung.

Verhandlungen zwischen Agenten basieren meist auf dem in Abschnitt 2.3.2 eingeführten CNP, beispielsweise erfolgreich angewendet in [159], [238], [239]. Lernende Steuerungs-Agenten nutzen, wie in Abschnitt 2.3.3 eingeführt, häufig das „trial-and-error“-Prinzip mithilfe von RL. Sie sind Fokus des in Kapitel 5 erarbeiteten lernenden Steuerungsansatzes. Agentenarchitekturen werden nach Ouelhadj und Petrovic [75] in autonome und Mediator-Architekturen eingeteilt. Erstere arbeiten mit vollständig autonom und heterarchisch organisierten Agenten, während in Mediator-Architekturen auch übergreifend koordinative Agenten im Einsatz sind. Leitao [88] spricht im gleichen Zusammenhang von folgenden Architekturen:

1. Rein **agentenbasierte** Architekturen,

2. **Holonische** Architekturen.

Wie in Abschnitt 2.3.2 eingeführt, nutzen holonische Architekturen wiederum die Agententechnologie. Hierbei kombiniert eine holonische Steuerung koordinativ-zentrale Tätigkeiten mit autonom-dezentral agierenden Agenten. Sie vereint zentrale mit dezentralen Steuerungseigenschaften [167].

Sowohl rein agentenbasierte als auch holonische Steuerungsarchitekturen wenden auktionsbasierte Verhandlungen zwischen Agenten bzw. Holonen für die Entscheidungsfindung an. Damit überschneiden sie sich mit der spieltheoretischen Organisation nach [101] und es kommt häufig das CNP zum Einsatz [75]. Eine klare Trennung der Lösungsfelder ist deshalb schwer möglich. Da Auktionen in beiden Architekturarten präsent sind und damit die spieltheoretische Organisation implizit vorhanden ist, konzentrieren sich die nächsten beiden Abschnitte auf ausgewählte dezentrale Steuerungsansätze rein agentenbasierter sowie holonenbasierter Art. Die Verhandlungen zwischen Agenten zur Entscheidungsfindung stehen hierbei im Vordergrund. Darauf basierend wird anschließend ein dezentraler Steuerungsansatz für die MP entwickelt, der die in Abschnitt 2.4.2 gestellten Anforderungen erfüllt. Einen ergänzenden Überblick zu den zahlreich entwickelten dezentralen Steuerungsmethoden geben u.a. [75], [88], [101], [124], [167], [168], [201], [237], [240].

4.1.2 Analyse ausgewählter dezentraler Steuerungsansätze

Die Analyse der bestehenden Literatur dezentraler Steuerungsansätze erfolgt anhand 10 ausgewählter und dem Steuerungsproblem der MP verwandter Steuerungslösungen. Die Auswahl orientiert sich an der Anwendbarkeit und Relevanz für die MP sowie an der Ausführlichkeit der Beschreibung der jeweiligen Lösungen. Die gewählten Ansätze sind den in Abschnitt 4.1.1 abgeleiteten Methoden der rein agentenbasierten und holonenbasierten Ansätze zuzuordnen. Diese beiden Lösungsfelder verschwimmen aufgrund ihrer im letzten Abschnitt beschriebenen engen Vernetzungen, u.a. durch die häufig auktionsbasierte Lösungsfindung, miteinander.

Agentenbasierte Ansätze: Multiagenten-Scheduling stützt sich meist auf verhandelnde Agenten mithilfe des CNPs. Deshalb steht die Kommunikation zwischen den Agenten im Fokus. Bussmann und Schild [241]–[243], Rajabinasab und Mansour [244], Badr [245], Wang et al. [127], [246], [247], Leusin et al. [248], [249] und Gehlhoff et al. [250], [251] stellen agentenbasierte Steuerungsansätze für flexible Produktionssysteme vor.

Zusammen mit Daimler entwickelten Bussmann und Schild [241]–[243] im Projekt *Production 2000+* ein flexibles Fertigungssystem für die Antriebsfertigung mit flexiblem Auftragsrouting. Analog zu dieser Arbeit ist dabei jede CNC-Maschine mit einem EP, einem Bearbeitungsplatz sowie einem AP ausgestattet, die nach dem FIFO-Prinzip funktionieren. Die entwickelte Steuerung nutzt ereignisorientiert stattfindende Auktionen. Die wesentlichen Steuerungsentscheidungen treffen hierbei Produkt-Agenten. Bei Auftragseinlastung bzw. nach jedem abgeschlossenen Produktionsschritt sendet der jeweilige Produkt-Agent eine Anfrage an die Maschinen zur Weiterbearbeitung. Zwischen den bietenden Maschinen entscheidet sich das Produkt primär auf Basis ihrer Pufferbelegung. Für jede Auktion kommt eine einfache Erstpreisauktion (engl.: first-price auction) auf Basis des CNPs zum Einsatz. Bei Störungen regelt sich das Produktionssystem über die Bevorzugung geringerer Pufferbelegungen von selbst, indem sich der Puffer einer gestörten Maschine füllt und die Produkte daher Alternativrouten suchen. Außerdem nimmt ein Maschinen-Agent parallel an mehreren Auktionen teil. Wie die Entscheidungen beispielsweise bei zwei gleichzeitig gewonnenen Auktionen einer Maschine koordiniert werden, wenn nur ein Platz im Puffer frei ist, bleibt offen. Nach der Entscheidung für eine Maschine werden die Produkt-Agenten vom AP der alten Maschine über ein Fließbandsystem mithilfe von Schiebeeinheiten auf dem möglichst kürzesten Weg zum EP der nächsten Maschine transportiert. Ein FTS kommt daher nicht zum Einsatz, auch Transportzeiten werden nicht in die Entscheidungen eingebunden. Die Beschreibung einer Beispielproduktion in [241] lässt außerdem darauf schließen, dass alle Maschinen eine identische Bearbeitungszeit anbieten. Damit fehlt die für die MP charakteristische Fertigungszeitspreizung. Unterschiedliche Fertigungszeiten könnten u.a. eine entscheidende Rolle für die durchgeführten Verhandlungen spielen. Außerdem ist aus den Arbeiten nicht ersichtlich, wie die Auftragseinlastung gesteuert wird. Zielkriterium der Ent-

scheidungsfindung mittels CNP ist der Produktionsdurchsatz, wie auch in dieser Arbeit. Hier erreichen die Autoren für ein gegebenes Produktionssystem von Daimler nahezu ein theoretisches Durchsatzmaximum sowie hohe Robustheit gegenüber Maschinenstörungen. Auch in einer praktischen Umsetzung bestätigen sich diese Ergebnisse.

In Rajabinasabs und Mansours [244] MAS für eine flexible Werkstattfertigung entscheiden Maschinen- und Auftrags-Agenten über das Auftragsrouting, das Stationsmanagement sowie das Störungsmanagement. Ein Manager-Agent verwaltet die entscheidungsrelevanten Informationen. Die Auftragseinlastung der Auftrags-Agenten wird durch stochastische Ankunftszeiten simuliert. Flottenmanagement findet keine Berücksichtigung. Die Agenten kommunizieren mithilfe eines dem CNP ähnelnden Protokolls. Das Auftragsrouting eines Auftrags-Agenten entscheidet sich wie bei Bussmann und Schild [242] Operation für Operation. Nach jeder Operation holt der entsprechende Auftrags-Agent hierzu Angebote der für die Bearbeitung infrage kommenden Maschinen ein. Befindet sich eine Maschine in einer Störung, nimmt sie anders als bei Bussmann und Schild an keiner derartigen Verhandlung teil. Bevorzugt werden wiederum Maschinen mit kürzeren Warteschlangen. Analog zur obigen Verhandlung agieren Maschinen zur Auswahl des nächsten Auftrags aus ihren Puffern. Hier gilt somit kein FIFO. Stattdessen holt der Maschinen-Agent bei Wiederverfügbarkeit Angebote der im Puffer liegenden Auftrags-Agenten ein. Er entscheidet sich bevorzugt für den Auftrag mit der höchsten verbleibenden Bearbeitungszeit und der geringsten Zeitspanne bis zum terminierten Fertigstellungszeitpunkt. Letztere Entscheidung ist in dieser Arbeit nicht notwendig, weil die Stationspuffer wie bei Bussmann und Schild nach dem FIFO-Prinzip arbeiten. Das beschriebene MAS schneidet für ein Beispiel mit acht Maschinen und sechs Produkttypen besser ab als fünf gängige Prioritätsregeln.

In ihrer Dissertation entwickelt Badr [245] ein MAS zur Steuerung eines flexiblen Fertigungssystems. Das MAS integriert ankommende Produktionsaufträge in den aktuellen Produktionsschedule. Hierzu verhandelt der das einzulastende Produkt repräsentierende Auftrags-Agent mit den Maschinen- und Fahrzeug-Agenten mithilfe eines CNPs über mehrere Hierarchieebenen [245, S. 62ff.]. Diese Verhandlung berücksichtigt alle durchzuführenden Operationen und Transporte des Auftrags bis zur Fertigstellung, was Badrs Steuerung von anderen Ansätzen unterscheidet, welche meist Operation für Operation vorgehen. Damit beinhaltet der aktuelle Produktionsschedule des flexiblen Fertigungssystems zu jeder Zeit alle Operationen und Transporte aller aktiven Produktionsaufträge [245, S. 71ff.]. Im Störungsfall werden betroffene Teilbereiche dieses Schedules neu verhandelt, sodass ein gültiger Gesamtschedule erhalten bleibt, wobei Effizienz und Stabilität des Schedules im Vordergrund stehen [245, S. 88ff.]. Das beschriebene Verfahren zielt auf eine Optimierung der Produktionsspanne aller Aufträge ab. Mit diesem Ansatz deckt Badr einen großen Teil der Anforderungen dieser Arbeit ab und fokussiert vor allem auch auf die Integration des Flottenmanagements. Allerdings bringt ein Gesamtschedule zwei wesentliche Nachteile mit sich. Erstens steigt die Rechenzeit in Abhängigkeit der Problemgröße bei der Schedule-Erstellung meist exponen-

tiell an, was die Echtzeitanforderung aus Abschnitt 2.4.2 gefährdet. Zweitens schränkt ein Gesamtschedule die Flexibilität ein, die ein MAS bietet. Ergeben sich in der Produktion unvorhergesehene Situationen durch Produktionsabweichungen (unabhängig von Stationsstörungen), für die ein anderer Schedule besser wäre, berücksichtigt der Gesamtschedule dies nicht, weil die Aufträge bereits vollständig zu Ende geplant sind. Auch in der Literatur haben sich deshalb situative MAS-Entscheidungen durchgesetzt. Die situative Agentenkooperation, wobei Steuerungsentscheidungen Operation für Operation getroffen werden, stellt für die MP daher die dezentrale Methode der Wahl dar. Dennoch bieten Badrs Arbeiten wertvolle Anhaltspunkte zur Konzeption eines MAS.

Die „Smart Factory“ ist das Kernthema der Arbeiten von Wang et al. [127], [246], [247]. In ihrem gesamtheitlichen Produktionskonzept berücksichtigen sie daher Industrie 4.0 Technologien wie CPS, CC und IdD (vgl. Abschnitt 2.3.1). Es resultiert eine realitätsnahe Implementierung einer „Smart Factory“-Steuerung. Die Steuerung basiert auf der Kommunikation von Echtzeitdaten zwischen als virtuelle Recheneinheiten implementierten Produktions-Agenten und Steuerungselementen. Kernstück der Steuerung ist ein CNP-basierter Verhandlungsmechanismus zwischen Produkt-Agenten, Maschinen-Agenten, Puffer-Agenten und Transport-Agenten. Der Produkt-Agent verhandelt seine Produktionsschritte mit den Maschinen bzw. lässt sich bei Notwendigkeit puffern. Für jeden neuen Bearbeitungsschritt startet der Agent daher eine neue Auktion. Zwischen den von Maschinen oder Puffern eingereichten Geboten entscheidet der Produkt-Agent regel- und prioritätsbasiert. Beispielsweise bevorzugt das Produkt kürzere Transportrouten. Den für den verhandelten Produktionsschritt notwendigen Transport organisiert das Produkt anschließend durch eine weitere Verhandlung mit den verfügbaren Transport-Agenten. Der Ansatz agiert demzufolge aus Produktsicht, und zwar aus der jeweiligen Sicht des auktionierenden Produkt-Agenten. Damit findet keine direkte Koordination zwischen mehreren Produkt-Agenten statt, obwohl sie um die Produktionskapazität der Maschinen konkurrieren. Außerdem bieten bei einer Auktion nur unbelegte Maschinen-Agenten mit den für die Produktbearbeitung passenden Fähigkeiten. Eine Maschine ist damit nicht in der Lage, schon frühzeitig an Auktionen teilzunehmen. Dies verhindert, dass beispielsweise bei Fertigstellung des aktuell bearbeiteten Produktes bereits ein neues Produkt zur Bearbeitung bereitsteht. Somit ist kein vorausschauendes und koordiniertes Entscheiden möglich. Darunter leidet die Produktionseffizienz. Anstatt die Produktionseffizienz zu maximieren, setzen die Autoren ihren Fokus auf eine zentral koordinierende Komponente zur Vermeidung von Stau- und Blockadesituationen. Diese Art der Koordination ist für das in Abschnitt 2.4.2 definierte Problem aktuell nicht relevant. Trotzdem bieten die realitätsnahe Implementierung sowie der dargestellte Verhandlungsmechanismus für Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement einige Anhaltspunkte, auf denen diese Arbeit aufbauen kann.

In [248], [249] erarbeiten Leusin et al. ein MAS zur Steuerung einer eher linienförmig organisierten Werkstattfertigung in Form eines CPPS (vgl. Abschnitt 2.3.1). Hierbei legen

die Autoren Wert auf eine realitätsnahe Implementierung. Sie berücksichtigen beispielsweise die Anbindung an ein ERP-System sowie die Kommunikation zwischen verschiedenen Agenten, beispielsweise mittels TCP/IP. Zu Simulationszwecken erfolgt die Implementierung der Produktionsentitäten außerdem auf unterschiedlichen Recheneinheiten. Die in der Produktion getroffenen Entscheidungen basieren auf den vorhandenen Echtzeitdaten des CPPS. Hierfür baut das MAS analog zu Badrs [245] Arbeit auf eine hierarchische Organisation. Auf der untersten Ebene wählen Maschinen-Agenten jeweils den nächsten zu bearbeitenden Produkt-Agenten aus ihrem Puffer mithilfe von Prioritätsregeln. Eine Ebene höher verwalten Koordinations-Maschinen-Agenten jeweils eine Gruppe von Maschinen-Agenten. Den Gesamtüberblick behält ein Agenten-Koordinator, der die Produkt-Agenten mithilfe der ihm zur Verfügung gestellten Produktionsinformationen den Maschinen-Agenten zuweist. Steuerungseingriffe auf dieser Ebene erfolgen ebenfalls mittels Prioritätsregeln. In der gegebenen Implementierung fehlt dem MAS eine Komponente, die mehrere Produkt-Agenten gleichzeitig in die Entscheidungsfindung einbezieht. Stattdessen trifft die Steuerung die Entscheidungen für Produkt-Agenten sequenziell und unabhängig voneinander. Vorgesehen, aber nicht implementiert, wird hier eine übergreifende KI, die derartige Koordinationsentscheidungen berücksichtigen könnte. In der aktuell dargestellten Implementierung steht daher mehr die realitätsnahe CPPS-Implementierung als die Erreichung einer möglichst hohen Lösungsgüte im Vordergrund. Die CPPS-Implementierung dient deshalb als Orientierung für das Konzept des dezentralen Steuerungsansatzes dieser Arbeit, weil sie die in Abschnitt 2.4.2 gestellten Anforderungen an Echtzeitfähigkeit und Datenverfügbarkeit erfüllt.

Kovalenko et al. [252], [253] stellen eine MAS-Architektur mit Fokus auf Produkt-Agenten für eine flexible Halbleiterfertigung vor. Jeder Produkt-Agent plant seine Route im Produktionssystem selbstständig in Absprache mit den Ressourcen. Zur Entscheidungsfindung verfügt jeder Produkt-Agent über eine Wissensbasis, eine Entscheidungseinheit und eine Kommunikationseinheit [254]. Hierbei orientieren sich Kovalenko et al. an [255]. Die Entscheidungseinheit ist dafür verantwortlich, auf Basis des lokalen Wissens in Absprache mit den Ressourcen-Agenten die richtigen Entscheidungen entsprechend der Ziele des Produkt-Agenten zu treffen. Für die Schedule-Berechnung des Produkt-Agenten kommt hierbei ein Dijkstra-Algorithmus zum Einsatz, welcher aus den zur Verfügung stehenden Aktionen den zeitlich kürzesten Weg für den Agenten sucht. Da wie bei Wang et al. [127], [246], [247] oder Leusin et al. [248], [249] jeder Produkt-Agent egoistisch agiert und sich weder mit anderen Produkt-Agenten abstimmt noch eine Optimierung zwischen unterschiedlichen Ressourcen stattfindet, ist die Lösungsgüte im Hinblick auf die Ressourcenauslastungen und den Durchsatz des Produktionssystems verbesserungswürdig. Hier zeigt sich, dass die Sicht des Produkt-Agenten eine gesamtheitliche Optimierung im Allgemeinen erschwert. Eigene Vorarbeiten der vorliegenden Arbeit bestätigen diesen Zusammenhang [32]. Aufgrund der egoistischen Sichtweise führt Kovalenko in [256, S. 67ff.] einen Koordinationsmechanismus zwischen Produkt- und Ressourcen-Agenten ein. Der Mechanismus verbessert die Lösungsgü-

te für ein eher kleines Produktionsbeispiel mit zwei Maschinen und zwei Transportrobotern. Jedoch betonen die Autoren, dass die verwendeten Evaluationsbeispiele höher zu skalieren sind, vor allem im Hinblick auf reale komplexe Halbleiterfertigungssysteme. Außerdem fehlt ein Vergleich zu alternativen Steuerungsmechanismen. Dieser Vergleich für die Steuerung einer MP ist Kern dieser Arbeit. Aufgrund der hohen Realitätsnähe bei der Implementierung ist das vorgestellte Framework trotzdem als gute Grundlage für Steuerungsalgorithmen mit Fokus auf Produkt-Agenten zu sehen. Eine detaillierte Zusammenfassung des Ansatzes findet sich in [256].

Gehlhoff und Fay [250] entwickeln ein CNP-basiertes MAS zur dezentralen Erstellung globaler Schedules. Der Schedule wird vor Produktionsstart von den Agenten verhandelt und dann anschließend zur Steuerung der Produktion verwendet. Wie bei Badr [245] verwaltet die Steuerung damit einen globalen Schedule. Hierzu baut sich der Plan auf, indem jeder Auftrags-Agent mit Maschinen, Fahrzeugen und Puffern verhandelt, um einen möglichst schnellen Weg durch das Produktionssystem zu finden. Die Aufträge werden hierfür nacheinander und jeweils bis zum Ende ihrer Bearbeitung eingeplant. Damit arbeitet auch dieser Ansatz egoistisch aus Auftragssicht. Zur Verbesserung der Produktionsspanne aller Aufträge könnte wie bei Kovalenko [256, S. 67ff.] eine parallele Koordination mehrerer Produktionsaufträge Abhilfe schaffen, anstatt sie nacheinander in den Schedule zu übernehmen. Der erstellte Schedule berücksichtigt Stationsmanagement und Auftragsrouting. In einer Erweiterung in [257] leitet das MAS automatisiert ab, welche Routingoptionen für die Produktionsschritte eines Auftrags infrage kommen. Dies geschieht fähigkeitsbasiert, ähnlich der Beschreibung aus Abschnitt 2.1.2 für die MP. Außerdem erweitern die Autoren das Steuerungskonzept in [251] um ein Störungsmanagement, welches simulativ getestet wird. Der global abgelegte Gesamtschedule wird hierfür im Störungsfall ab dem Auftreten der Störung neu verhandelt. Die Neuberechnung findet allerdings nicht im Echtzeitbereich statt, sondern dauert unter Berücksichtigung der verwendeten Rechenressourcen mehrere Sekunden. Als Vorteil zeigen die Evaluationsexperimente in [258] jedoch, dass die Rechenzeiten der Scheduling-Verhandlungen in Abhängigkeit der teilnehmenden Produktionsentitäten linear skalieren, wobei die Komplexität der Kommunikation anderer Ansätze exponentiell wächst. Dies ist vor allem für globale Schedules von elementarer Bedeutung.

Holonenbasierte Ansätze: Die zwei bekanntesten holonischen Steuerungsarchitekturen PROSA und ADACOR präsentieren van Brussel et al. [164] sowie Leitao und Restivo [165]. Beide dienen als Rahmenwerke zur Implementierung holonischer Steuerungen. Als Rahmenwerke werden sie in Bezug auf die Anforderungen der Arbeit in Abschnitt 4.1.3 für jedes Kriterium als „teilweise berücksichtigt“ eingestuft. Erol et al. [259] wenden PROSA zur dynamischen Steuerung eines flexiblen Fertigungssystems an. Sie beschreiben im Gegensatz zu den Rahmenwerken eine konkrete Steuerungslösung.

Als PROSA stellen van Brussel et al. [164] eine holonische Steuerungsarchitektur für Pro-

duktionssysteme vor, welche in der Lage ist, vollständig dezentral zu agieren. Eine optionale zentrale Komponente ermöglicht globale Optimierung, dient allerdings als Ratgeber, nicht als endgültiger Entscheider. Vier Holonenklassen prägen die Architektur: „product“ (Prod-H), „order“ (Ord-H), „resource“ (Res-H) und „staff holon“ (Staff-H). Ein Prod-H repräsentiert das Produkt- und Prozesswissen, um ein Produkt zu fertigen. Ein Ord-H steht für ein eingelastetes Produkt bzw. einen eingelasteten Produktionsauftrag, welcher seine Produktionsschritte durchläuft. Jede Ressource im System wird hingegen durch ein Res-H verwaltet, welches seine Fähigkeiten und seinen Status speichert sowie Produktionskapazität zur Verfügung stellt. Ein optionales Staff-H unterstützt durch zentrales Wissen die anderen Holone. Im Störungsfall tendieren Prod-H, Ord-H und Res-H dazu, die zentralen Ratschläge zu ignorieren, um autark und autonom mit Störfaktoren wie Maschinenausfällen umzugehen. Eine nicht näher spezifizierte Metasteuerung verwaltet das Umschaltspiel zwischen zentral und dezentral. Eine dynamische Gestaltung des Dezentral-Zentral-Wechselspiels beschreiben beispielsweise Borangiu et al. [170]. Die Gestaltung der Entscheidungsfindung innerhalb der Architektur basiert auf der Zusammenarbeit der Holone, wird allerdings nicht näher beschrieben. Damit bietet die PROSA-Architektur ein Rahmenwerk für die Entwicklung dezentraler Steuerungsansätze. Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement sind entsprechend selbst auszuarbeiten.

Mit ADACOR führen Leitao und Restivo [165] eine holonische Steuerungsarchitektur für flexible Werkstattfertigungssysteme ein. Wie bei PROSA verändert sich die Holonenorganisation im Störungsfall von einer hierarchisch geprägten zu einer dezentral geprägten. Unabhängig von Störungen ist das generelle Ziel der Steuerung, so dezentral wie möglich und so zentral bzw. hierarchisch wie nötig zu agieren. Hierfür stellen die Autoren vier unterschiedliche Holonenklassen vor: „product“ (Prod-H), „task“ (Task-H), „operational“ (Op-H) und „supervisor holon“ (Sup-H). Damit wählen Leitao und Restivo eine zu PROSA ähnliche Struktur. Prod-H (Prod-H in PROSA), Task-H (Ord-H in PROSA) und Op-H (Res-H in PROSA) repräsentieren Produkte, Aufträge und Ressourcen. Der wesentliche Unterschied liegt im Sup-H, welches im Gegensatz zu PROSAs Staff-H nicht optional ist, sondern periodisch und ereignisgesteuert die dezentral zwischen Op-Hs und Task-Hs verhandelten Schedules optimiert. Sup-Hs übernehmen damit die Rolle der globalen Optimierung einer Holonengruppe und organisieren die Holarchie hierarchisch. Insgesamt lebt ADACOR damit von der CNP-basierten Kommunikation zwischen Holonen und der resultierenden adaptiven Koordination der Produktion. Ein Autonomiefaktor steuert je nach Produktionssituation, ob die Steuerung in Form der Op-Hs eher dezentral agiert oder von Sup-Hs hierarchisch organisiert wird. Das Verhalten der Holone wird durch einen Satz von Regeln bestimmt. Leitao und Restivo zeigen in ihrer Evaluation die Funktionalität und den Mehrwert von ADACOR basierend auf [260]. Wie bei PROSA gilt: ADACOR stellt ein Rahmenwerk zur dezentralen Produktionssteuerung dar, in das sich die fünf Entscheidungskategorien dieser Arbeit integrieren lassen. Ausführlichere Informationen finden sich in [261]. Eine beispielhafte Er-

weiterung von ADACOR im Hinblick auf die adaptive Selbstorganisation zeigen Barbosa et al. [169].

Erol et al. [259] konzipieren ein MAS zur dynamischen Echtzeitsteuerung eines flexiblen Fertigungssystems auf Basis der PROSA-Architektur. Die durchzuführenden Operationen eines Auftrags werden durch einzelne Operations-Agenten abgebildet, welche in der vordefinierten Fertigungsreihenfolge Schritt für Schritt aktiviert bzw. deaktiviert werden. Andere Ansätze wählen hier einen Auftrags- oder Produkt-Agenten, der die Operationen direkt ohne zusätzliche Agenten verwaltet, was die Anzahl der Agenten reduziert und zum gleichen Ergebnis führt. Jeder aktive Operations-Agent agiert autonom und organisiert seinen Transport durch ein FTF und seine Bearbeitung an der entsprechenden Maschine. Der Transport entscheidet sich durch eine Auktion mittels CNP. Mit seiner geschätzten Abholzeit fragt der Operations-Agent hierzu bei den FTF-Agenten an und wählt das beste Transportangebot aus. Hierbei bestimmt die voraussichtliche Ankunftszeit an der nächsten Station die Güte der Transportangebote der FTF-Agenten. Bei Ankunft an der geplanten Maschine wird der aktive Operations-Agent des Auftrags in den dortigen Puffer aufgenommen. Die Maschinen-Agenten wiederum bestimmen die Bearbeitungsreihenfolge konkurrierender Operations-Agenten aus ihrem Puffer heuristisch mithilfe der geringsten Bearbeitungszeit (Stationsmanagement). Analog zu Rajabinasabs und Mansours [244] Ansatz widerspricht dies der in dieser Arbeit angenommenen FIFO-Reihenfolge innerhalb von Arbeitsstationen. In einer Erweiterung des Ansatzes in [262] berücksichtigen die Autoren zusätzlich das Auftragsrouting, wobei ein zum Flottenmanagement analoger Verhandlungsmechanismus zum Einsatz kommt. Auftragseinlastung und Störungsmanagement werden nicht explizit betrachtet. In ihrer Evaluation vergleichen Erol et al. ihr MAS mit dokumentierten Offline-Benchmarks und erreichen i.d.R. gute Schedules, schneiden allerdings mehrheitlich schlechter ab als die Optimallösungen der Benchmark-Probleme.

4.1.3 Schlussfolgerungen für die Entwicklung der dezentralen Steuerung

Tabelle 4.1 analysiert die in Abschnitt 4.1.2 beschriebenen dezentralen Steuerungsansätze anhand der Anforderungen der MP (vgl. Abschnitt 2.4.2) an die fünf Entscheidungskategorien Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement (1.1-1.5) sowie anhand der Anforderungen an die Durchsatzmaximierung bzw. Minimierung der Produktionsspanne (2), die Echtzeitfähigkeit (3), die Datenverfügbarkeit (4) und die Generalisierbarkeit (5) der Ansätze.

Im Gegensatz zu klassischen zentralen Ansätzen (vgl. Abschnitt 3.1.3) legen dezentrale MAS den Fokus auch auf Echtzeitfähigkeit und Datenverfügbarkeit. Diese Kriterien sind vor allem aufgrund der in der Realität notwendigen Kommunikation zwischen Produktions-Agenten relevant. Echtzeitfähigkeit ist hierbei in den meisten Fällen gegeben, weil die Agen-

Tabelle 4.1: Überblick zu ausgewählten dezentralen Steuerungsmethoden in mit der MP verwandten Produktionssystemarten.

Legende: ● berücksichtigt ◐ teilweise berücksichtigt ○ nicht berücksichtigt, nicht ersichtlich	1.1 **Auftragseinlastung**	1.2 **Stationsmanagement**	1.3 **Auftragsrouting**	1.4 **Flottenmanagement**	1.5 **Störungsmanagement**	2 **Durchsatzmaximierung**	3 **Echtzeitfähigkeit**	4 **Datenverfügbarkeit**	5 **Generalisierbarkeit**
Agentenbasierte Ansätze									
Bussmann und Schild (2000, 2001, 2007)	○	●	●	◐	●	●	●	●	○
Rajabinasab und Mansour (2011)	○	●	●	○	●	○	◐	◐	○
Badr (2011)	○	●	●	●	●	●	◐	◐	◐
Wang et al. (2016, 2016b, 2017)	○	●	●	●	◐	◐	●	●	○
Leusin et al. (2018, 2018b)	◐	●	●	○	◐	○	●	●	○
Kovalenko et al. (2019, 2019b), Kovalenko (2020)	○	●	●	●	●	◐	●	●	◐
Gehlhoff und Fay (2020), Gehlhoff et al. (2021)	○	●	●	●	◐	◐	◐	◐	○
Holonenbasierte Ansätze									
Van Brussel et al. (1998)	◐	◐	◐	◐	◐	◐	◐	◐	◐
Leitao und Restivo (2006)	◐	◐	◐	◐	◐	◐	◐	◐	◐
Erol et al. (2012)	○	●	●	●	○	●	◐	○	●

ten eher lokale Steuerungsprobleme lösen, als globale Pläne zu berechnen. Die Lösungsgüte dieser lokalen Teilprobleme wird auch hier häufig mit einem Durchsatzmaximum oder einer minimalen Produktionsspanne bestimmt. In Kombination mit dem Konzept der VIBN aus Abschnitt 2.3.4 legen z.B. die realitätsnahen Implementierungen von Kovalenko et al. und Leusin et al. sowie die holonenbasierten Architekturen PROSA und ADACOR einen soliden Grundstein für die spätere Implementierung des MAS. In Bezug auf die fünf Entscheidungskategorien fällt auf, dass Stationsmanagement, Auftragsrouting, aber vor allem auch Flotten- und Störungsmanagement ein elementarer Bestandteil der Steuerungskonzepte sind. Die Auftragseinlastung spielt hingegen selten eine Rolle. Meist simulieren die Autoren mit stochastisch verteilten Ankunftszeiten der Produktionsaufträge. Für die spätere Konzeptionierung der Auftragseinlastung können Löddings Kriterien für Einlastungszeitpunkte aus Abschnitt 2.2.1 dienen. Auch die Performanz der Steuerungsansätze bei unterschiedlichen Produktionskonfigurationen wird spärlich nachgewiesen. Ausschließlich Erol et al. analysieren ein breites Spektrum an Produktionskonfigurationen und demonstrieren damit die Generalisierbarkeit der Steuerung. Insgesamt erschließt keiner der Ansätze die in dieser Arbeit untersuchte Breite des Steuerungsproblems und erfüllt alle Anforderungen.

Am engsten an den Anforderungen dieser Arbeit sind die Arbeiten von Bussmann und Schild [241]–[243], Badr [245], Wang et al. [127], [246], [247] sowie Kovalenko et al. [252], [253], [256]. Badrs Steuerung verwaltet einen globalen Schedule, der agentenbasiert aufgebaut und ereignisgesteuert adaptiert wird, was die Echtzeitfähigkeit sowie die MAS-Flexibilität im Sinne dieser Arbeit einschränkt. Der Ansatz wird konzeptionell daher nicht in Betracht gezogen. Bussmann und Schild sowie Wang et al. arbeiten hingegen mit lokalen CNP-basierten Auktionen, welche die Entscheidungen Operation für Operation herbeiführen und keinen globalen Plan erstellen. Damit ist Echtzeitfähigkeit gegeben. In beiden Ansätzen stoßen Produkt-Agenten die Auktionen an und entscheiden sich egoistisch für ihr Optimum. Eine Koordination zwischen mehreren Produkten findet daher nicht statt. Dies scheint auf die Lösungsgüte in Bussmanns und Schilds Arbeit keinen negativen Einfluss zu haben, was möglicherweise an der spezifischen Transportorganisation über ein flexibles Fließbandsystem liegt. Wang et al. fehlt die Koordination konkurrierender Produkte für die Maximierung der Lösungsgüte hingegen. Das Koordinationsproblem zwischen mehreren Produkten zeigt sich auch in den Arbeiten von Kovalenko et al., wobei egoistische Produktentscheidungen nicht zum Optimum führen. Außerdem bestätigen eigene Untersuchungen die Wichtigkeit der Koordination zwischen Produkten [32].

Aufgrund des durchsatzoptimalen CNP-Ansatzes orientiert sich diese Arbeit konzeptionell an Bussmann und Schild [241]–[243]. Wegen der problembedingten Transportunterschiede (Fließband vs. FTS) im Produktionssystem wird zusätzlich die in vielen Implementierungen fehlende Koordination zwischen konkurrierenden Produkten eingeführt. Dies wird durch eine Auktionsumkehr erreicht. Nicht die Produkte starten Auktionen, sondern modulare Stationen stellen ihre Kapazität zur Auktion. In diesen Auktionen können dann konkurrierende Produkte teilnehmen, was einen breiteren Blick auf das Gesamtproblem ermöglicht und die Produkte gesamtheitlicher koordiniert. Die lokalen Auktionsentscheidungen sind dabei auf das globale Ziel der Durchsatzmaximierung der MP auszulegen.

Bevor Abschnitt 4.3 das entwickelte Gesamtkonzept einer dezentralen Steuerung erläutert, geht Abschnitt 4.2 auf das CNP und die damit verbundene Auktion als Kernmethode des Konzepts ein.

4.2 Methode: Contract Net Protocol und Erstpreisauktion

Nach Fatima et al. [263, S. 1] ist der Zweck einer Verhandlung, eine Übereinkunft in der Gegenwart gegenläufiger Ziele zu erreichen. Wie Abschnitt 4.1 zeigt, hat sich für die Übereinkunft mehrerer Agenten im dynamischen Produktions-Scheduling das von der FIPA [264] standardisierte (vgl. Abschnitt 2.3.2) und von Smith [161] eingeführte CNP als Verhandlungsprotokoll durchgesetzt. Hierzu nutzen die verhandelnden Agenten das Protokoll situa-

tiv, wenn Entscheidungen der Ressourcenallokation anstehen [263, S. 145].

Diese dynamisch zu treffenden Entscheidungen basieren deshalb auf im MAS vorhandenen Echtzeitinformationen zur aktuellen Produktionssituation. Demnach wird das in Abbildung 4.1 dargestellte CNP durch einen die Verhandlung initiierenden Agenten angestoßen. Hierzu definiert der initiierende Agent die zu lösende Aufgabe und schickt sie an m potenzielle Auktionsteilnehmer, die ebenfalls Agenten sind. Bis zu einer definierten Deadline T_{dead} haben die Auktionsteilnehmer Zeit, entweder an der Auktion teilzunehmen und für die Aufgabe zu bieten oder die Teilnahme abzulehnen. Die Deadline stellt sicher, dass der initiierende Agent eine endliche Zeit auf potenzielle Gebote wartet. Nach der Deadline eintreffende Gebote werden vom Initiator nicht berücksichtigt. Von den j antwortenden Agenten lehnen $q \leq j$ ab, wohingegen die Gebote der $n = j - q$ bietenden Agenten nun vom Initiator der Auktion bewertet werden können. Nach der Bewertung lehnt der Auktionsinitiator $k \leq n$ Gebote ab und akzeptiert die anderen $l = n - k$ oder er lehnt alle Gebote ab, weil keines der Gebote seinen Anforderungen entspricht. Die l akzeptierten Teilnehmer können die an sie gestellte Aufgabe nun ausführen und abschließend das Ergebnis zur Verfügung stellen [264].

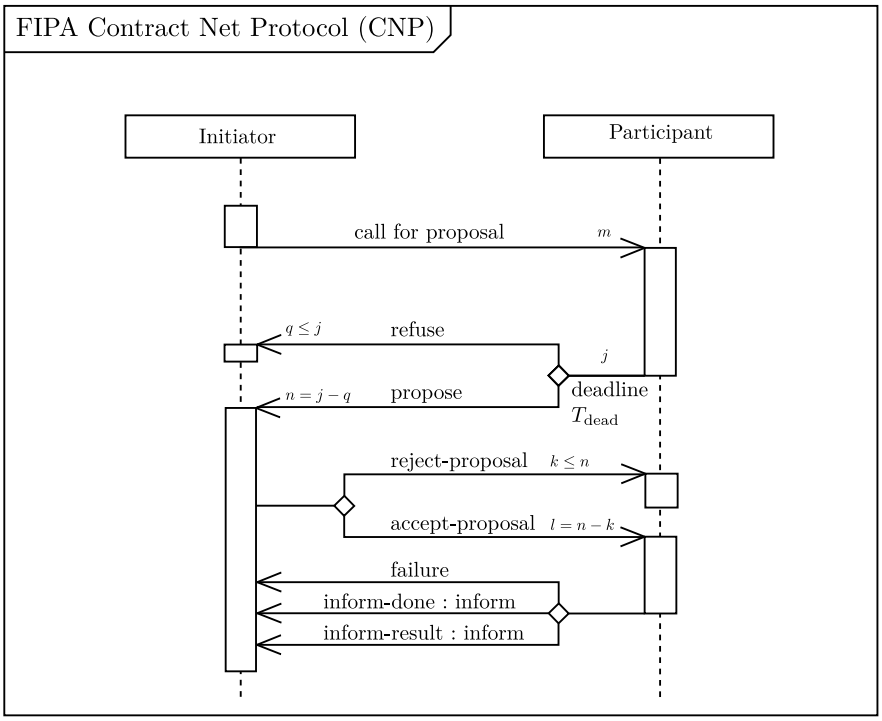


Abbildung 4.1: FIPA CNP, Quelle: Eigene Darstellung in Anlehnung an [161], [264].

Das CNP offenbart hierbei weder die Gestaltung der Gebote der bietenden Agenten noch die Entscheidungsfindung des Auktionators. Hierfür bieten Bussmann und Schild [242] Anhaltspunkte. Sie setzen die Entscheidungsfindung im CNP erfolgreich mithilfe einer einfachen Erstpreisauktion um (vgl. Abschnitt 4.1). In [241] begründen sie diese Wahl damit, dass die an der Produktionssteuerung beteiligten Agenten ein globales Produktionsziel ver-

folgen, welches vom Designer der Steuerung vorgegeben wird. Die Agenten werden deshalb so konzipiert, dass sie für das Gemeinwohl handeln und wahrheitsgemäß bieten. Diese Zusammenhänge gelten auch in dieser Arbeit. Die Produktionssteuerung strebt das globale Ziel einer Durchsatzmaximierung an (vgl. Abschnitt 2.4.2). Die Agenten sollen entsprechend dieses Ziels handeln.

Bei Bussmann und Schild [242] bieten Maschinen für einen Produktionsvorgang eines Produktes. Es steht ein einzelnes Gut, der Produktionsvorgang, zur Auktion. Daher handelt es sich um eine sogenannte Eingutauktion. In den später konzipierten Auktionen dieser Arbeit handelt es sich ebenfalls um eine Eingutauktion. Hierbei betrifft das Gut die von einer Maschine angebotene Produktionskapazität.

In einer Eingutauktion hat jeder Agent i der n Bieter eine nicht-negative Bewertung $v_i \geq 0$ für das auktionierte Gut. Sie ist privat und den anderen Bietern unbekannt. Die Bewertung v_i definiert, was der Bieter maximal bezahlen möchte bzw. wie viel ihm das Gut wert ist. Die quasilineare Nutzenfunktion eines gewinnenden Bieters berechnet sich zu $u_i = v_i - p$, wobei p der Preis ist, zu dem der Agent das Gut ersteht. Alle anderen Bieter haben keinen Nutzen von der Auktion. Ihr Nutzen ist 0 [265, S. 11f.]. Die folgenden Absätze gehen von einer Eingutauktion aus.

Eine Auktion kann verdeckt („sealed bid“) oder auch offen („open cry“) durchgeführt werden. Bei offenen Auktionen sehen die Bieter auch die Gebote der anderen. Bei verdeckten Auktionen übergeben Bieter ihr Gebot privat [263, S. 142]. Das CNP läuft verdeckt ab, weil die Agenten ihr Gebot direkt an den Auktionator kommunizieren. Damit handelt es sich um eine verdeckte Eingutauktion. Eine verdeckte Eingutauktion hat folgendes Format: Erstens werden die Gebote privat abgegeben. Zweitens entscheidet der Auktionator, wer die Auktion gewinnt. Drittens legt der Auktionator den Preis p für das Gut fest [263, S. 142]. Die folgenden Ausführungen konzentrieren sich auf die verdeckte Eingutauktion.

Der zweite Schritt der verdeckten Auktion ist meist klar definiert. Der höchste Bieter ersteigert das auktionierte Gut. Unterschiedliche Auktionstypen unterscheiden sich hauptsächlich im dritten Schritt, zu welchem Preis das Gut verkauft wird [265, S. 12]. In einer Erstpreisauktion zahlt der Gewinner der Auktion sein abgegebenes Gebot b_i: $p = b_i$. Hingegen zahlt der Gewinner in einer Zweitpreisauktion das zweithöchste Gebot. Die Zweitpreisauktion entspricht der von ebay angewendeten Auktionsart, sie wird auch als Vickrey-Auktion bezeichnet [265, S. 12ff.], [263, S. 142]. Weitere Auktionsarten sind beispielsweise die englische oder die holländische Auktion. Mehr zu diesen Auktionsarten und u.a. zu Mehrgutauktionen ist beispielsweise in [266, S. 163-177] nachzulesen.

Bussmann und Schild [242] wählen eine Erstpreisauktion. In verdeckten Erstpreisauktionen mit strategischen Bietern versuchen die Bieter ihren Nutzen u_i zu maximieren. Da die Bieter jedoch keine Informationen zu den Geboten der anderen Bieter erhalten, fällt die Auswahl der Höhe des Gebots schwer [265, S. 14]. Bei Bussmann und Schild [242] verfolgt die Produktionssteuerung das globale Ziel der Durchsatzmaximierung. Bieter müssen sich

daher nicht strategisch verhalten, weil sie ein gemeinsames Ziel verfolgen. Damit wird obiges Problem verdeckter Erstpreisauktionen obsolet. Die Auktionsteilnehmer bieten ihre Bewertung $b_i = v_i$ und haben dabei keine Kosten: $p = 0$. Der Nutzen ist: $u_i = v_i - 0 = v_i$. Der Auktionsteilnehmer mit der höchsten Bewertung v_i erhält den Zuschlag, weil es dem sozialen Gemeinwohl aller Bieter dient und dadurch den Gesamtnutzen maximiert. Die Auktion optimiert das soziale Gemeinwohl, weil sie $\sum_{i=1}^{n} v_i x_i$ maximiert, wobei $x_i = 1$ beim Gewinn der Auktion und andernfalls $x_i = 0$ gilt. Da es sich um eine Eingutauktion handelt, gilt: $\sum_{i=1}^{n} x_i \leq 1$ [265, S. 15-25]. Die Auktion ohne strategische Bieter benötigt dabei keine komplexe Bezahlungsregel, wie sie zum Beispiel Roughgarden [265, S. 26] für verdeckte Erstpreisauktionen einführt.

Abgeleitet von [265, S. 26] besteht die verdeckte Eingutauktion damit aus zwei Schritten, wohingegen der dritte Schritt der Bezahlungsregel aufgrund obiger Erklärung wegfällt:

1. **Gebote sammeln**: Sammle Gebote b_i der n bietenden Agenten in Form eines Vektors $\boldsymbol{b} = (b_1, \ldots, b_n)$.

2. **Allokationsregel**: Wähle eine **Allokation** des zu versteigerten Guts $\boldsymbol{x}(\boldsymbol{b}) \in X \subseteq \mathbb{R}^n$ als Funktion der Gebote $\boldsymbol{b}$, wobei X die Menge aller validen Lösungen darstellt.

Für das Sammeln der Gebote müssen die agentenspezifischen Bewertungen v_i sowie die darauf basierenden Gebote b_i wahrheitsgemäß situativ berechnet werden. Die Allokationsregel bestimmt dann anhand der n Gebote, welcher Agent die Auktion gewinnt.

Die in diesem Abschnitt eingeführte Formalisierung einer verdeckten Erstpreisauktion ohne strategische Bieter, in welcher ein Gut versteigert wird, dient als Grundlage zur Gestaltung der CNP-Verhandlungen im MAS, um die Entscheidungskategorien Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement in einer MP zu lösen. Weiterführende Informationen zu Auktionen und Verhandlungen sind beispielsweise in [263] oder in [147] nachzulesen.

4.3 Konzeptionierung des auktionsbasierten Multiagentensystems

Abschnitt 4.1.3 leitet aus dem bestehenden Stand der Forschung ein Grobkonzept zur dezentralen MAS-Steuerung der MP ab, vor allem auf Basis von [242]. Abschnitt 4.2 stellt anschließend die Methode zur Koordination der wesentlichen Steuerungsentscheidungen im MAS vor und führt hierfür das CNP sowie damit verbundene Auktionen ein. Darauf basierend fasst Abbildung 4.2 das auf das Steuerungsproblem dieser Arbeit angepasste Feinkonzept des dezentralen Steuerungsansatzes zusammen und stellt überblicksweise die Lösung der fünf Entscheidungskategorien aus Abschnitt 2.2.2 dar. Teile dieses Ansatzes wurden in [32] und [33] veröffentlicht.

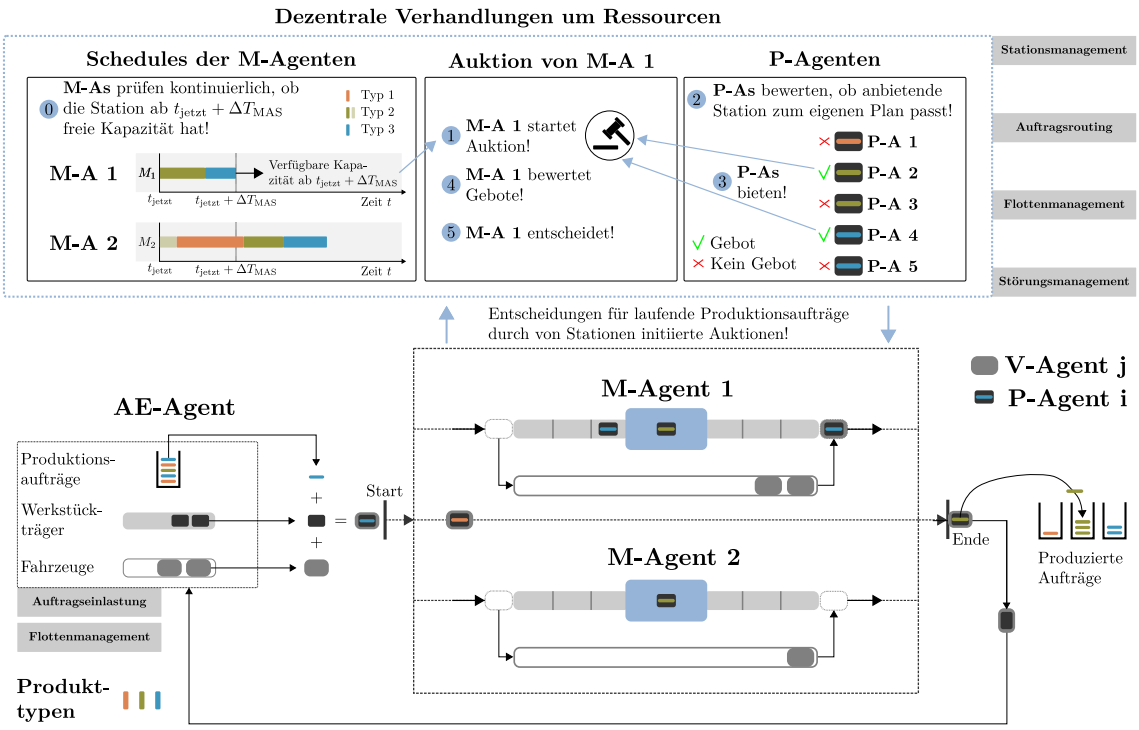


Abbildung 4.2: Auktionsbasiertes MAS zur Steuerung der MP: Ein AE-Agent koordiniert die Auftragseinlastung, mithilfe von CNP-basierten Erstpreisauktionen verhandeln P-Agenten (auch P-A im Bild) mit M-Agenten (auch M-A im Bild) um Produktionskapazität, V-Agenten transportieren P-Agenten von Station zu Station.

Die Auftragseinlastung wird von einem Auftragseinlastungs(AE)-Agenten organisiert. Dieser weist den zu startenden Produkttypen einen WT sowie ein FTF und damit einen Fahrzeug(V)-Agenten zu. Da der WT nur Mittel zum Zweck für den Transport eines Produkts ist, führt dieser Abschnitt keinen WT-Agenten ein, sondern konzentriert sich in der Folge auf das Produkt, welches der WT durch die Produktion trägt. Die Festlegung der tatsächlichen Einlastungszeitpunkte der Produkte sowie das Stationsmanagement, das Auftragsrouting, das Flottenmanagement und das Störungsmanagement der eingelasteten Produkte erfolgen durch die in Abschnitt 4.2 eingeführte Erstpreisauktion basierend auf dem CNP. Hierbei verhandeln Maschinen(M)- und Produkt(P)-Agenten. Im Gegensatz zu [242] initiieren nicht die P-Agenten die Auktionen, sondern die M-Agenten. Auktionen aus Stationssicht ermöglichen eine Koordination zwischen konkurrierenden P-Agenten. M-Agenten prüfen hierfür kontinuierlich durch einen Blick um die Zeitspanne ΔT_{MAS} in die Zukunft, ob sie freie Kapazität in einer Auktion anbieten können. Ist dies der Fall, starten sie eine Erstpreisauktion und sammeln Gebote ein. P-Agenten bieten in gestarteten Auktionen, falls die anbietende Station in den eigenen Produktionsplan passt. Der M-Agent bewertet die eingehenden Gebote und entscheidet sich für einen P-Agenten anhand einer Allokationsregel. Dadurch findet die besagte Koordination konkurrierender Produkte statt, welche einigen Ansätzen aus Abschnitt 4.1 fehlt. Unter Berücksichtigung der voraussichtlichen Ankunftszeit wird der Bearbeitungsvor-

gang des P-Agenten in den Schedule des M-Agenten eingetragen. In der voraussichtlichen Ankunftszeit findet u.a. die Transportzeit und somit ein Teil des Flottenmanagements Berücksichtigung. Für den Zeitraum einer Stationsstörung startet der betroffene M-Agent keine Auktionen (Störungsmanagement). Erst nach der Störung wird er wieder aktiv. Die regelmäßig von M-Agenten initiierten Auktionen zerlegen das globale Steuerungsproblem in kleinere lokale Probleme. Die Steuerung verzichtet dadurch auf den gesamtheitlichen Blick auf die Entscheidungskategorien, findet jedoch, den MAS-Eigenschaften aus Abschnitt 2.3.2 entsprechend, flexible und robuste Lösungen für die Abläufe in der MP.

Zur Erläuterung des in Abbildung 4.2 visualisierten dezentralen Steuerungskonzepts betrachtet Abschnitt 4.3.1 zuerst die Auftragseinlastung, bevor die Abschnitte 4.3.2 und 4.3.3 auf die weiteren Entscheidungskategorien eingehen. Abschließend fasst Abschnitt 4.4 die wesentlichen Aspekte des dezentralen Steuerungskonzepts zusammen und vergleicht den Ansatz mit den in Abschnitt 2.4.2 definierten Anforderungen der MP-Steuerung.

4.3.1 Auftragseinlastung

Die Auftragseinlastung im MAS erfolgt durch den AE-Agenten. Das Prinzip ähnelt der Einlastung der zentralen Steuerung aus Abschnitt 3.3.1. Der Unterschied liegt in der Festlegung der Einlastungszeitpunkte. Der Vollständigkeit und Lesbarkeit halber werden die überschneidenden Aspekte hier erneut beschrieben.

Der AE-Agent geht nach den in Abschnitt 2.4.2 getroffenen Annahmen von einem vordefinierten Produktmix aus. Dieser legt die Anzahl der zu fertigenden Produkte pro Produkttyp fest und in welchem Verhältnis sie einzulasten sind. Zusammen mit der empfohlenen Periodizität der einzulastenden Produkte bildet der Produktmix das Produktionsprogramm und damit eine Liste abzuarbeitender Produktionsaufträge. Zu treffen sind Entscheidungen über die Zuweisung von WTs und FTFs sowie über die Zeitpunkte der Einlastung der einzelnen Produktionsaufträge. Eine erfolgreiche Einlastung erfordert deshalb die Verfügbarkeit eines WTs und eines FTFs, auf denen das Produkt im System gestartet werden kann. Damit betrifft die Einlastung auch einen Teil des Flottenmanagements. Die Anzahl verfügbarer WTs bzw. FTFs ($n_{\mathrm{WT}} = n_V$, Annahme in Abschnitt 2.4.2) begrenzt den Produktionsbestand in der MP zu: $PB_{max} = n_{\mathrm{WT}} = n_V$. Hierfür bietet sich eine Orientierung an der in Abschnitt 2.2.1 vorgestellten einfachen Conwip-Steuerung an, welche einen neuen Auftrag einlastet, wenn gilt: $PB < PB_{max}$ [74, S. 373-385]. Mit diesem Hintergrund, ergänzt mit einigen problemspezifischen Erweiterungen, folgt die dezentrale Auftragseinlastung dem Produktionsbestand und basiert auf zwei Prinzipien:

1. **Initiale Einlastung**: Im Initialzustand der Produktion stehen n_{WT} WTs und n_V Fahrzeuge hintereinander in der Quelle der MP bereit. Die periodische Auftragseinlastung beginnt in der vordefinierten Auftragsreihenfolge des Produktionsprogramms

unter Verwendung der verfügbaren WTs und FTFs. Der AE-Agent organisiert die Rahmenbedingungen der Auftragseinlastung. Er initiiert die einzulastenden P-Agenten und weist ihnen die WTs und FTFs in der vorhandenen Reihenfolge zu. Damit werden also die verfügbaren n_{WT} WT-FTF-Paare (WT_1/V_1, WT_2/V_2 etc.) den ersten Produktionsaufträgen des Produktionsprogramms zugeordnet. Der tatsächliche Zeitpunkt der Auftragseinlastung S_{J_i} des Auftrags J_i entspricht dem Zeitpunkt, zu dem der J_i repräsentierende P-Agent eine Auktion für seinen ersten Produktionsschritt gewinnt. Solange der einzulastende P-Agent keine Auktion gewinnt, wartet er in der Quelle der MP und es kann auch kein weiterer P-Agent gestartet werden. Dies stellt die geplante Auftragsreihenfolge im Produktionsprogramm sicher.

2. **Bestandsgeregelte Einlastung**: Sind alle WTs und FTFs in der MP im Einsatz, gilt: $PB = PB_{max}$. Der AE-Agent wartet dann auf ein verfügbar werdendes WT-FTF-Paar durch die Fertigstellung eines Produktes, wodurch PB unter die Bestandsgrenze PB_{max} fällt. Unter Beibehaltung des Produktionsprogramms erfolgen dann die Bereitstellung eines neuen Produktionsauftrages und die damit verbundene Initiierung des P-Agenten, sobald der WT und das FTF in der Quelle der MP ankommen und die Zuweisung erfolgt ist. Wiederum entspricht der letztendliche Einlastungszeitpunkt S_{J_i} dem Zeitpunkt einer gewonnenen Auktion aus Sicht des P-Agenten.

Diese Art der Einlastung setzt das geplante Produktionsprogramm mit der Losgröße eins um. Außerdem erfolgt die feste Verknüpfung von Produktionsauftrag, WT und FTF für die abzuarbeitenden Produktionsprozesse des Produktes, was ebenfalls als Annahme in Abschnitt 2.4.2 eingeführt wird. Die verfügbaren WTs und FTFs sind dem zweiten Prinzip der Einlastung entsprechend dauerhaft im Einsatz.

Auf die Einlastung folgt die Koordination der Produktionsabläufe der Aufträge und damit das Stationsmanagement, das Auftragsrouting und der zweite Teil des Flottenmanagements.

4.3.2 Stationsmanagement, Auftragsrouting und Flottenmanagement

Stationsmanagement und Auftragsrouting entscheiden sich durch von M-Agenten initiierte Auktionen. Auch die Transporte und damit das Flottenmanagement werden hierbei berücksichtigt. Abschnitt 4.3.1 führt außerdem ein, dass sich der Zeitpunkt der Auftragseinlastung durch eine Auktion entscheidet. Jede Auktion basiert auf dem in Abschnitt 4.2 eingeführten CNP und folgt dem Prinzip einer Erstpreisauktion. Dabei wird angenommen, dass alle Bieter in den Auktionen dasselbe globale Produktionsziel der Durchsatzmaximierung verfolgen und damit nicht strategisch agieren. Jede Auktion versteigert ein Gut: die Produktionskapazität einer Station. Die Gebote werden mittels CNP verdeckt übermittelt. Es handelt sich also

um eine verdeckte Eingutauktion. Abbildung 4.3 stellt das Sequenzdiagramm der Auktion dar. Die folgenden Absätze beschreiben die nummerierten Vorgänge im CNP.

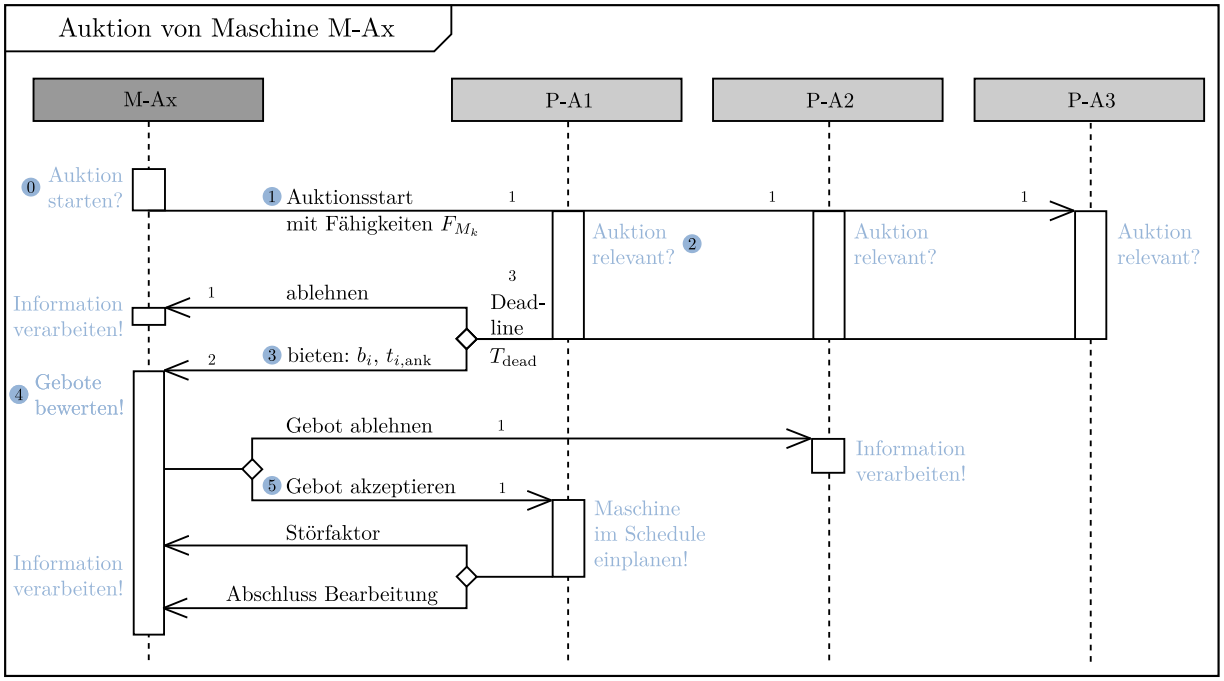


Abbildung 4.3: Ablauf einer von einem M-Agenten (M-A im Bild) gestarteten Auktion mit beispielhaft drei teilnehmenden P-Agenten (P-A im Bild) nach dem CNP.

Schritte 0 und 1: Jeder M-Agent verwaltet seinen individuellen Schedule und schaut kontinuierlich ΔT_{MAS} voraus. Dabei sucht er nach freier Kapazität, die er zur Bearbeitung von Produktionsvorgängen anbieten kann. Hierbei bezeichnet t_{frei} den Zeitpunkt, an dem die Station voraussichtlich alle geplanten Operationen beendet haben wird. Freie Zeitfenster zwischen zwei Operationen werden nicht angeboten. Der M-Agent startet eine Auktion, wenn gilt:

1. Innerhalb des Vorausschauhorizonts ΔT_{MAS} wird die Station vollständig frei: $t_{\text{frei}} - t_{\text{jetzt}} \leq \Delta T_{\text{MAS}}$.
2. Der M-Agent ist aktuell verfügbar. Er unterliegt demnach keiner Störung.
3. Der M-Agent hat nicht bereits eine Auktion gestartet. Jeder M-Agent veranstaltet zu jeder Zeit maximal eine Auktion.

Die Pufferkapazitäten betrachtet der M-Agent hierbei nicht, weil Fahrzeuge auch vor der Station warten können (vgl. Abschnitt 2.4.2) und damit als zusätzlicher Puffer fungieren. Die Nachricht des Auktionsstarts wird an alle P-Agenten gleichzeitig übermittelt, dargestellt durch den durchgezogenen Pfeil. Sie beinhaltet die angebotenen Fähigkeiten F_{M_k} der auktionierenden Station M_k.

Schritt 2: Nach Start der Auktion sammelt der M-Agent Gebote $\boldsymbol{b} = (b_1, \ldots, b_n)$ von n P-Agenten ein. Entsprechende Gebote müssen vor Ablauf der Deadline T_{dead} eintreffen. Alle P-Agenten bewerten, ob die gestartete Auktion von Relevanz für sie ist. Grundsätzlich bietet ein P-Agent immer nur für seinen nächsten Produktionsvorgang. Die Entscheidung, an welchen Stationen er für seinen nächsten Produktionsschritt bietet, beeinflusst sein Auftragsrouting. Er gibt ein Gebot ab, wenn folgende Bedingungen erfüllt sind:

1. **Position des P-Agenten**: Der P-Agent befindet sich entweder in der Quelle, in Bearbeitung an einer Station oder im AP einer Station.
   - In Quelle: Der P-Agent steht zur Einlastung bereit und wartet auf einen Auktionsgewinn (vgl. Abschnitt 4.3.1). Hierbei kann sich immer nur ein P-Agent gleichzeitig in der Quelle befinden.
   - An Station in Bearbeitung: Der P-Agent befindet sich in Bearbeitung an einer Station. Die bearbeitende Station darf sich hierbei nicht in Störung befinden.
   - An Station im AP: Der P-Agent befindet sich im AP einer Station. Abbildung 4.4 visualisiert die beiden letzteren Fälle.

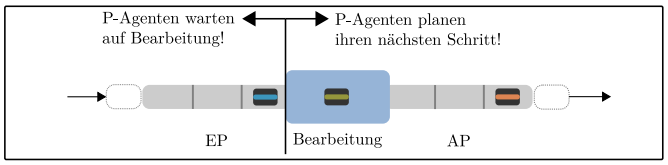


Abbildung 4.4: Befinden sich P-Agenten an einer Station, ist eine Grundvoraussetzung für eine Teilnahme an einer Auktion, dass sich der P-Agent in Bearbeitung oder im AP befindet.

2. **Berechtigung erteilt**: Der M-Agent, bei dem sich der P-Agent aktuell befindet, hat dem P-Agenten eine Berechtigung zur Teilnahme an Auktionen erteilt. Den zugrunde liegenden Mechanismus beschreibt Abbildung 4.5. Er vermeidet Blockaden und stellt sicher, dass P-Agenten an Stationen in der korrekten FIFO-Reihenfolge bieten.

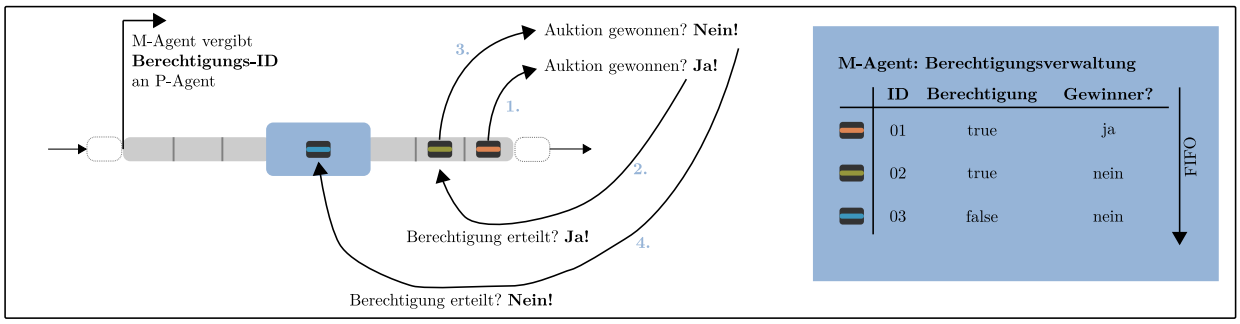


Abbildung 4.5: P-Agenten dürfen an Auktionen nur teilnehmen, falls alle an der aktuellen Station vorgelagerten P-Agenten bereits eine Auktion gewonnen haben, wozu der jeweilige M-Agent Berechtigungen erteilt.

3. **Auktionsgewinn ausständig**: Der P-Agent hat für den anstehenden Produktionsvorgang O_{ipl} noch keine Auktion gewonnen.

4. **Fähigkeitenabgleich**: Der auktionierende M-Agent der Station M_k bietet Fähigkeiten F_{M_k}, die zur Bearbeitung der nächsten Operation O_{ipl} eines P-Agenten i gebraucht werden: $F_{ipl} \subseteq F_{M_k}$. Diese fähigkeitsbasierte Denkweise ist auch in den Arbeiten von Wang et al. [127], [246], [247] zur Steuerung einer „Smart Factory“ zu finden.

Schritt 3: Sind alle Bedingungen erfüllt, berechnet der P-Agent i ein Gebot b_i für die anbietende Station. Hierbei streben alle Bieter die Durchsatzoptimierung an, arbeiten also mit demselben globalen Produktionsziel. Das Gebot b_i repräsentiert in einer derartigen Auktion die Wichtigkeit ($b_i = v_i$ in Abschnitt 4.2) eines Auktionsgewinns aus Sicht des P-Agenten. Die Gebotsgestaltung zielt auf eine Durchlaufzeitminimierung ab, was indirekt nach dem Gesetz von Little (vgl. (2.5)) zur Durchsatzoptimierung führt. b_i berechnet sich zu:

$$b_i = \delta \cdot t_{\text{warte},i} + \frac{t_{\text{sys},i}}{t_{\text{proc,total},i}} \cdot 1s, \tag{4.1}$$

wobei δ die beiden Terme gewichtet. $t_{\text{warte},i}$ beschreibt die Aufenthaltsdauer von P-Agent i im AP. Die Wartezeit beginnt beim Übergang vom Bearbeitungsplatz in den AP. Befindet sich der P-Agent noch in Bearbeitung und noch nicht im AP, gilt: $t_{\text{warte},i} = 0$. Damit versucht ein P-Agent so wenig Zeit wie möglich mit Warten zu verlieren, um seine Durchlaufzeit zu minimieren. Befindet sich das Produkt im AP, steigt der Term von Sekunde zu Sekunde an und agiert im Vergleich zum zweiten Term, vor allem zu Beginn der Auftragsproduktion sehr dominant. Der zweite Term im Gebot setzt die bisherige Aufenthaltsdauer im Produktionssystem $t_{\text{sys},i} = t_{\text{jetzt}} - S_{J_i}$ mit der gesamt abzuarbeitenden Prozesszeit $t_{\text{proc,total},i} = \frac{\sum_{p=1}^{n_P} \sum_{l=1}^{n_O} t_{ipl}}{n_P}$ des P-Agenten i ins Verhältnis. Da es sich um ein Verhältnis handelt, muss der Term mit $1s$ multipliziert werden, damit die Einheiten stimmen. Das Verhältnis erhöht das Gebot eines P-Agenten umso deutlicher, je länger er sich im System befindet. Damit werden in Auktionen neben wartenden Produkten solche bevorzugt, die sich näher an der Fertigstellung befinden. Dies sorgt einerseits dafür, dass ebenfalls die Gesamtdurchlaufzeit des P-Agenten minimiert wird und andererseits, dass durch die beschleunigte Fertigstellung der Produkte neue Produkte eingelastet werden können. Im Vergleich zu wartenden Produkten hat dieser zweite Term zu Beginn der Auftragsproduktion wenig bis keinen Einfluss auf die Entscheidungen, weil er im Vergleich zur Wartezeit $t_{\text{warte},i}$, welche im Sekundenbereich liegt, Werte von $\frac{t_{\text{sys},i}}{t_{\text{proc,total},i}} < 1$ annimmt. Jedoch entscheidet der zweite Term zwischen Produkten, die sich noch in Bearbeitung befinden ($t_{\text{warte},i} = 0$). Für die vorliegende Arbeit hat sich eine Gewichtung der beiden Terme mit $\delta = 1$ als empirisch zuverlässig gute und robuste Lösung herauskristallisiert, um den Durchsatz zu optimieren. Im Allgemeinen lässt sich festhalten: Je länger ein Produkt im System ist, desto mehr Einfluss hat der zweite Term der Gebotsbe-

rechnung. Bieten jedoch wartende Aufträge in der Auktion mit, entscheidet der erste Term meist die Auktion. Ein Sonderfall ist die Auftragseinlastung. Liegt ein P-Agent zur Einlastung bereit, bietet er nur auf Basis seiner Wartezeit $t_{\text{warte},i}$, da der offizielle Zeitpunkt der Einlastung (vgl. Abschnitt 4.3.1) noch nicht erfolgt ist. Damit gilt für diesen Fall: $t_{\text{sys},i} = 0$.

Zusätzlich zum Gebot b_i berechnet jeder P-Agent seine voraussichtliche Ankunftszeit $t_{i,\text{ank}}$ als optimistische Schätzung für eine durchzuführende Operation O_{ipl} zu:

$$t_{i,\text{ank}} = \begin{cases} t_{\text{jetzt}} + t_{\text{TR},ipl} + t_{\text{EP},min} & \text{für } J_i \text{ in Quelle,} \\ t_{\text{jetzt}} + t_{ip(l-1),\text{Rest}} + t_{\text{AP},min} + t_{\text{TR},ipl} + t_{\text{EP},min} & \text{für } J_i \text{ in Bearbeitung,} \\ t_{\text{jetzt}} + t_{\text{AP,Rest}} + t_{\text{TR},ipl} + t_{\text{EP},min} & \text{für } J_i \text{ in AP.} \end{cases} \tag{4.2}$$

Im ersten Fall befindet sich der bietende P-Agent i in der Quelle. Er ist direkt abfahrbereit, wodurch sich die geschätzte Ankunftszeit vom jetzigen Zeitpunkt t_{jetzt} mithilfe der Transportzeit $t_{\text{TR},ipl}$ zur anbietenden Station und der dortigen fließbandbedingten Mindest-Transportzeit im EP $t_{\text{EP},min}$ berechnen lässt. Im zweiten Fall befindet sich Agent i in Bearbeitung. Zusätzlich zum ersten Fall müssen somit die verbleibende Bearbeitungszeit $t_{ip(l-1),\text{Rest}}$ sowie die darauffolgende fließbandbedingte Mindest-Transportzeit im AP $t_{\text{AP},min}$ Berücksichtigung finden. Im dritten Fall ist die Bearbeitung bereits abgeschlossen und es fließt lediglich die verbleibende Förderzeit im AP $t_{\text{AP,Rest}}$ mit ein. Die drei Schätzungen berechnen die frühestmögliche Ankunftszeit an der anbietenden Station. Damit repräsentieren sie jeweils optimistische Schätzungen.

Am Ende übermittelt ein P-Agent sein Gebot b_i zusammen mit der voraussichtlichen Ankunftszeit $t_{i,\text{ank}}$ an den auktionierenden M-Agenten.

Schritt 4: Nachdem der M-Agent die Gebote b_i erhalten hat, bewertet er sie, indem er für jeden bietenden Auftrag Leerlaufkosten c_i in Abhängigkeit von der geschätzten Ankunftszeit des Auftrags $t_{i,\text{ank}}$ berechnet. Aufgrund der berücksichtigten Transportzeit spielt somit das Flottenmanagement eine Rolle. Auf Basis der optimistischen Schätzung der Ankunftszeit zeigen die Leerlaufkosten an, wie lange der M-Agent bei Annahme des Gebots mindestens leer stehen würde. Da alle P-Agenten optimistisch schätzen, ist die Vergleichbarkeit zwischen den Geboten gegeben. Die Leerlaufkosten berechnen sich zu:

$$c_i = \begin{cases} t_{i,\text{ank}} - t_{\text{frei}} & \text{für } t_{i,\text{ank}} \geq t_{\text{frei}}, \\ 0 & \text{für } t_{i,\text{ank}} < t_{\text{frei}}. \end{cases} \tag{4.3}$$

Je später ein P-Agent ankommt, desto größer sind die Leerlaufkosten. Die Station berechnet die Kosten aus der Differenz der Ankunftszeit $t_{i,\text{ank}}$ sowie der Startzeit des in der Auktion angebotenen Zeitslots t_{frei}. Damit maximiert eine Station ihre Auslastung. Kommt ein Produkt früher als t_{frei} an, berechnet sich c_i zu 0, weil die Station keine Kapazität verliert. Jeder

M-Agent strebt also eine Balance aus hoher Auslastung (c_i) und möglichst zeiteffizienten Transportrouten (b_i) an. Letzteres gilt, weil die P-Agenten zur individuellen Durchlaufzeitminimierung mit steigenden Wartezeiten in APs sowie mit steigender Aufenthaltsdauer in der MP höher bieten.

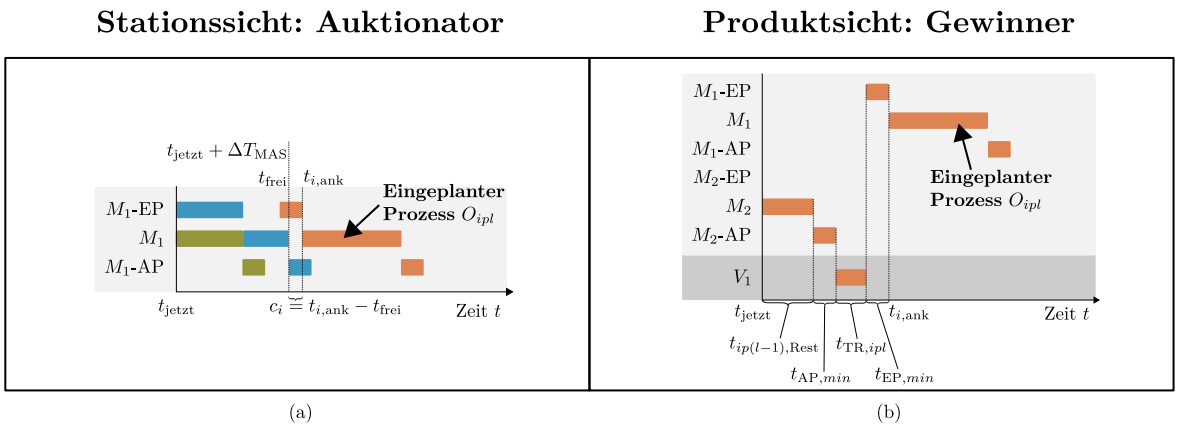
Die Kombination der Gebote b_i mit den Leerlaufkosten c_i zielt damit insgesamt auf die Durchsatzmaximierung unter möglichst effizienter Stationsnutzung ab. Am Ende gewinnt der P-Agent die Auktion, der (4.4) maximiert. Die Allokationsregel der Erstpreisauktion ist damit definiert als:

$$\max_{i \in J}(b_i - c_i). \tag{4.4}$$

Haben beispielsweise $n = 3$ Bieter i in der Auktion geboten und Bieter $i = 2$ gewinnt nach (4.4) die Auktion, ergibt sich die Allokation zu $\boldsymbol{x}(\boldsymbol{b}) = (0, 1, 0)$ (vgl. Abschnitt 4.2). Eine Bezahlung ist nach Abschnitt 4.2 nicht erforderlich. Diese Allokationsentscheidung bestimmt über Stationsmanagement und Auftragsrouting sowie, wenn der Gewinner ein einzulastender Auftrag ist, über den Zeitpunkt der Auftragseinlastung. Das Stationsmanagement legt der auktionierende M-Agent selbst fest, indem er einen neuen P-Agenten in den individuellen Maschinenschedule einplant. Das Auftragsrouting des P-Agenten entscheidet sich hingegen Produktionsschritt für Produktionsschritt, je nachdem an welchen Stationen der Agent die jeweilige Auktion für seinen nächsten Produktionsschritt gewinnt.

Schritt 5: Ist die Entscheidung gefallen, tragen sich sowohl der M-Agent als Auktionator als auch der gewinnende P-Agent den geplanten Bearbeitungsvorgang in ihre individuellen Schedules ein. Abbildung 4.6 stellt dies beispielhaft dar. In Abbildung 4.6a baut der M-Agent der Station M_1 den orange gefärbten Prozess in seinen bestehenden Schedule ein. Zu beachten sind auch die dargestellten Verbleibzeiten im EP und im AP der Station. Das blau gefärbte Produkt muss beispielsweise im EP warten, bis der braun gefärbte die Bearbeitung beendet hat. Generell sind die Mindest-Transportzeiten im EP und AP zu beachten, welche durch die Bewegung des WTs auf dem Fließband zustande kommen. In Abbildung 4.6b baut der gewinnende P-Agent den orange gefärbten Prozess in seinen individuellen Schedule ein. Hierbei ist auch zu erkennen, dass sich der P-Agent aktuell an Station M_2 in Bearbeitung befindet und anschließend zu M_1 mittels FTF V_1 transportiert wird.

Gewinnt ein P-Agent in mehreren Auktionen gleichzeitig, was z.B. möglich ist, wenn der nächste geplante Prozess von gleichzeitig auktionierenden Stationen durchgeführt werden kann, akzeptiert der P-Agent die zeitlich zuerst ankommende Gewinnermeldung mittels CNP und sagt der zweiten Auktion ab. Da dieses Szenario unwahrscheinlich ist, kann diese Vereinfachung getroffen werden. Der betroffene M-Agent, dem der Gewinner absagt, muss demnach den bereits im individuellen Schedule eingetragenen Prozess wieder löschen und startet daraufhin eine neue Auktion.



(a) (b)

Abbildung 4.6: Der Bearbeitungsvorgang des gewinnenden P-Agenten wird sowohl in den Schedule der auktionierenden Station (a) als auch in den Schedule des gewinnenden P-Agenten (b) eingetragen.

Ähnliche Schedule-Updates sind bei Störsituationen durchzuführen. Der nächste Abschnitt beschreibt den Umgang des MAS mit entsprechenden Stationsstörungen.

4.3.3 Störungsmanagement

Für ein effektives Störungsmanagement sind folgende erweiterte MAS-Funktionen notwendig:

- Die erste Funktion beschreibt bereits Abschnitt 4.3.2. Unterliegt ein M-Agent einer Störung und ist dadurch nicht verfügbar, startet er keine Auktionen mittels CNP. Dies verbindet den vorliegenden Ansatz mit der Implementierung von [244], unterscheidet ihn jedoch von [242], bei dem auch gestörte Maschinen an Auktionen teilnehmen. Im vorliegenden Ansatz ist dies nicht möglich, da ein M-Agent um ΔT_{MAS} in seinem Schedule vorausschaut. Dies ist im Störungsfall nicht möglich, weil das Ende der Störung nicht vorausgesagt werden kann. Mögliche Nachteile dieser Implementierung werden durch den letzten Stichpunkt dieser Aufzählung kompensiert.

- P-Agenten, die sich gerade in Bearbeitung auf einer gestörten Station befinden, bieten in keinen Auktionen mit, da sie ihre potenzielle Ankunftszeit $t_{i,\text{ank}}$ nicht realistisch schätzen können.

- Hat ein direkt von einer Störung betroffener P-Agent bereits vor Eintritt der Störung eine Auktion für den nächsten Produktionsschritt gewonnen, sagt er in Analogie zum Gewinn zweier paralleler Auktionen aus Abschnitt 4.3.2 der betroffenen Station ab. Der betroffene M-Agent löscht daraufhin den bereits eingetragenen Prozess aus seinem individuellen Schedule. Dies stellt sicher, dass die Kapazität anderen P-Agenten angeboten werden kann. Auch der P-Agent löscht den bereits gewonnenen nächsten Prozess aus seinem Schedule. Sobald die ihn betreffende Störung behoben ist, nimmt er wieder an Auktionen teil.

- P-Agenten, die wiederum an der aktuell gestörten Station schon eingeplant sind, bleiben auch eingeplant. Dies stellt sicher, dass bei Wiederanlauf der Station Produkte zur Bearbeitung bereitliegen, was mögliche Nachteile des obigen ersten Stichpunktes kompensiert. Mögliche längere Wartezeiten bei P-Agenten, die noch einen Alternativweg verfolgen könnten, nimmt das MAS aufgrund der Vorteile in Kauf.

Diese Erweiterungen ermöglichen einen unkomplizierten und effektiven Umgang mit Stationsstörungen.

4.4 Zusammenfassung und Einordnung des Ansatzes

Abschnitt 4.1 legt den grundlegenden Stand der Forschung dezentraler Steuerungsansätze dar, auf dessen Grundlage in Abschnitt 4.1.3 ein Grobkonzept abgeleitet wird. Anschließend beschreibt Abschnitt 4.2 die Grundlagen zu Verhandlungen zwischen Agenten als Methode für die dezentrale Entscheidungsfindung. Abschnitt 4.3 nutzt diese Methodik zur Beschreibung des Feinkonzepts des verhandlungsbasierten MAS, wobei M-Agenten ihre Kapazität zur Auktion anbieten und P-Agenten für diese Kapazität bieten. Ein auktionierender M-Agent schaut hierbei ΔT_{MAS} voraus, um vorhandene Kapazität anzubieten. Die folgenden Stichpunkte ordnen den dezentralen Ansatz in die in Abschnitt 2.4.2 gestellten Anforderungen an eine MP-Steuerung ein. Die einzelnen Punkte beschreiben hierbei entweder, wie die bereits beschriebenen Inhalte die Anforderungen adressieren, oder wie sie in der späteren Implementierung in Abschnitt 6.2.3 umzusetzen sind:

- **Auftragseinlastung**: Die Auftragseinlastung orientiert sich am Conwip-Prinzip. Abschnitt 4.3.1 beschreibt die konzeptionellen Zusammenhänge. Der Zeitpunkt der Einlastung ergibt sich aus dem Gewinn einer Auktion. Außerdem weist der AE-Agent einem einzulastenden Produkt ein FTF und einen WT zu, welche das Produkt für sämtliche Produktionsoperationen begleiten.

- **Stationsmanagement, Auftragsrouting und Flottenmanagement**: Die an die Einlastung anschließenden Entscheidungen zu Stationsmanagement, Auftragsrouting und Flottenmanagement sind Ergebnis der Verhandlungen zwischen Kapazität anbietenden M-Agenten und Kapazität suchenden P-Agenten. Die Inhalte hierzu finden sich in Abschnitt 4.3.2.

- **Störungsmanagement**: Das Störungsmanagement erfolgt durch erweiterte MAS-Funktionen, welche in Abschnitt 4.3.3 nachzulesen sind. Gestörte Stationen bieten ihre Kapazität für den Zeitraum der Störung nicht an. Von einer Störung betroffene P-Agenten sagen bereits für zukünftige Produktionsschritte gewonnene Auktionen ab, um die Stationskapazität anderen P-Agenten zur Verfügung zu stellen.

- **Durchsatzmaximierung**: Die Durchsatzmaximierung wird durch die Gestaltung der Auktionen in Abschnitt 4.3.2 erreicht, welche auf eine Durchlaufzeitminimierung ausgelegt ist, was wiederum nach dem Gesetz von Little zum Durchsatzmaximum führt.

- **Echtzeitfähigkeit**: Echtzeitfähigkeit garantiert die dezentrale Problemlösung, indem lokale Probleme in Form von Auktionen zu lösen sind, anstatt das globale Scheduling-Problem zu lösen. Hierzu wird die Deadline T_{dead} für gestartete Auktionen auf $1s$ gesetzt, um der Anforderung nach Echtzeitfähigkeit aus Abschnitt 2.4.2 zu entsprechen. Da die Kommunikation und die zugehörigen Berechnungen der Agenten im ms-Bereich stattfinden, ist dies legitim.

- **Datenverfügbarkeit**: Datenverfügbarkeit zeigt sich später in der beschriebenen Implementierung in Abschnitt 6.2.3. Für den dezentralen Steuerungsansatz ist hierfür eine dezentrale Datenablage auf den einzelnen Agenten notwendig. Außerdem ist dafür zu sorgen, diese Daten bzw. das damit verbundene Wissen der Agenten entsprechend der Produktionsereignisse aktuell zu halten.

- **Generalisierbarkeit**: Generalisierbarkeit auf unterschiedliche Produktionskonfigurationen im Rahmen der in Abschnitt 2.4.2 definierten Systemgrenze zeigt später Abschnitt 6.5. Folgende Punkte des Steuerungskonzepts ermöglichen derartige Generalisierungsexperimente: Zum einen ist die Auftragseinlastung flexibel, weil sie sich auf einen gegebenen Produktmix einstellt. Zum anderen ermöglichen die Auktionen, in denen M-Agenten spezifische Fähigkeiten zur Bearbeitung von Produkten anbieten, dass P-Agenten sich anhand ihrer benötigten Fähigkeiten ihren Weg durch die MP suchen. Hierbei können M-Agenten mit beliebigen Fähigkeiten ausgestattet werden.

Offen bleiben die Fragen zur Parametereinstellung und Implementierung. Der dominante Parameter des dezentralen Steuerungsansatzes ist die Vorausschau ΔT_{MAS} eines M-Agenten, um freie Kapazität im individuellen Schedule zu identifizieren. Diese Frage untersucht Abschnitt 6.3.2. Das Implementierungskonzept als zweite offene Frage ist hingegen Thema in Abschnitt 6.2.3. Dieser Abschnitt beschreibt eine realitätsnahe Implementierung, die die Erfüllung der Anforderungen an eine realitätsnahe MP-Steuerung sicherstellt.

Der entwickelte dezentrale Ansatz deckt die zweite der drei Steuerungsarchitekturen ab, welche die vorliegende Arbeit betrachtet. Das nächste Kapitel konzipiert einen lernenden Ansatz.

5 Lernender Ansatz: Deep Reinforcement Learning-Agent

Dieses Kapitel erarbeitet einen lernenden Ansatz zur Steuerung der MP in Form eines zentral lernenden Agenten und leistet damit einen Beitrag zur zweiten Forschungsfrage aus Abschnitt 1.2. Zuerst betrachtet Abschnitt 5.1 den bestehenden Stand der Forschung lernender Steuerungsansätze und schlussfolgert ein Grobkonzept. Anschließend erläutert Abschnitt 5.2 Proximal Policy Optimization (PPO) aus dem DRL-Bereich als verwendete Kernmethode zur Entscheidungsfindung im lernenden Steuerungskonzept. Abschnitt 5.3 beschreibt im Folgenden das entwickelte Feinkonzept des lernenden PPO-Agenten, bevor Abschnitt 5.4 den Ansatz zusammenfasst und mit den Anforderungen aus Abschnitt 2.4.2 abgleicht. Die DRL-Literatur ist hierbei sehr stark von englischen Begriffen geprägt. Um die Konsistenz mit der Literatur sicherzustellen, werden einige der englischen Begriffe übernommen.

5.1 Stand der Forschung lernender Steuerungsansätze

Dieser Abschnitt fasst den Stand der Forschung für lernende Steuerungsansätze in Bezug auf die MP zusammen. Abschnitt 5.1.1 verschafft hierzu einen Überblick zu den in der Literatur verwendeten Lösungsmethoden. Abschnitt 5.1.2 analysiert anschließend ausgewählte lernende Steuerungsansätze. Abschließend fasst Abschnitt 5.1.3 die untersuchten Ansätze zusammen, ordnet sie in die Arbeit ein und zieht Schlussfolgerungen für den in Abschnitt 5.3 entwickelten lernenden Steuerungsansatz.

5.1.1 Überblick zu lernenden Steuerungsansätzen

Abschnitt 2.3.3 leitet RL als geeignete Methode ab, um lernfähige Agenten und damit lernende Steuerungsansätze für die MP umzusetzen. Dabei erfolgt eine Einteilung in zentrale lernende Ansätze, wenn ein einzelner Agent beteiligt ist und dezentrale lernende Ansätze, wenn es sich um ein lernendes MAS handelt. Der Lernprozess basiert auf dem Sammeln von Erfahrungen in der Umgebung der Agenten.

Abhängig von der Speicherung der gesammelten Erfahrungen können RL-Methoden in tabellenbasierte [178, S. 23ff.] und in DRL-Methoden [178, S. 195ff.] eingeteilt werden. Tabellenbasierte Verfahren speichern gesammelte Erfahrungen in Tabellenform. DRL-Methoden

verwenden hierfür ein NN und arbeiten approximativ. Letztere werden eingesetzt, wenn die Problemkomplexität zu groß wird, um die gelernten Erfahrungen in Tabellen zu verarbeiten. Der limitierende Faktor ist hierbei die Rechenkapazität bzw. der Arbeitsspeicher [178, S. 195].

Bekannte tabellenbasierte Ansätze sind z.B. Sarsa und Q-learning. Beide Ansätze speichern die Werte aller Erfahrungen (State-Action-Kombinationen) in einer sogenannten „Q-Table“. Durch das Sammeln von Erfahrungen nähern sich die Algorithmen den optimalen Werten Schritt für Schritt an, wodurch die Q-Table das optimale Verhalten in der Agentenumgebung abbildet [178, S. 129, S. 131]. Auch für Produktionssteuerungsprobleme finden derartige tabellenbasierte Ansätze Anwendung, wie beispielsweise in [267]–[270]. Hierbei adressieren die Autoren entweder kleine Problemgrößen oder sie adressieren das Skalierbarkeitsproblem tabellenbasierter Ansätze, indem beispielsweise Clustering-Methoden zur Verkleinerung des Zustandsraums verwendet werden. Derartige Clustering-Methoden finden auch bei DRL-Ansätzen Anwendung, wenn komplexe Zustandsräume zu kontrollieren sind [271, S. 93ff.]. Aufgrund der Komplexität des in der vorliegenden Arbeit betrachteten Problems und der oben eingeführten Limitierung tabellenbasierter Ansätze bei höheren Komplexitäten konzentriert sich diese Arbeit auf DRL-Algorithmen.

Morales [179, S. 340] teilt DRL-Algorithmen in value-basierte, policy-basierte und Actor-Critic-Methoden ein. Value-basierte Methoden approximieren Value-Funktionen, welche die Werte verschiedener States (Zustände) und Actions (Aktionen) über die Rewards (Belohnungssignal) berechnen, wovon eine Policy abgeleitet wird. Policy-basierte Algorithmen überspringen diesen Zwischenschritt. Sie approximieren direkt die Policy, nach der der Agent mit der Umgebung interagiert. Actor-Critic-Methoden kombinieren diese beiden DRL-Arten und approximieren sowohl Value-Funktionen als auch direkt die Policy. Im Gegensatz zu Morales sehen Sutton und Barto [178, S. 331] Actor-Critic-Methoden als policy-basierte Methoden an. Die vorliegende Arbeit verwendet jedoch Morales’ Notation und spricht von value-basierten, policy-basierten und Actor-Critic-Methoden.

Die wohl bekannteste value-basierte DRL-Methode ist Deep Q-learning, wobei sogenannte Deep Q-Networks (DQN) zur Approximation einer Q-Table zum Einsatz kommen [176], [272]. Eine weit verbreitete policy-basierte Methode ist Trust Region Policy Optimization (TRPO) [273]. Unter den Actor-Critic-Methoden ist PPO ein vielversprechender Ansatz [274]. Abschnitt 5.1.3 entscheidet, welche dieser DRL-Methoden die vorliegende Arbeit anwendet. Das Verständnis der DRL-Methodik sowie der soeben verwendeten Begriffe schärft das später folgende Methodenkapitel in Abschnitt 5.2.

Einen Überblick zu lernenden Steuerungsansätzen bieten z.B. [101], [275]–[277]. Wie der nächste Abschnitt zeigt, finden hierbei sowohl DQN als auch TRPO und PPO in der Literatur zur Steuerung von Produktionssystemen Anwendung.

5.1.2 Analyse ausgewählter lernender Steuerungsansätze

Die Analyse der bestehenden Literatur lernender Steuerungsansätze erfolgt anhand 10 ausgewählter und dem Steuerungsproblem der MP verwandter Steuerungslösungen. Die Auswahl orientiert sich an der Anwendbarkeit und Relevanz für die MP sowie an der Ausführlichkeit der Beschreibung der jeweiligen Lösungen. Die gewählten Ansätze sind den in Abschnitt 5.1.1 abgeleiteten zentralen und dezentralen DRL-Ansätzen zuzuordnen.

Zentrale DRL-Ansätze: Zentrale DRL-Ansätze steuern die Produktion mit einem einzigen lernenden Agenten. Derartige Ansätze verwenden Mahadevan und Theocharous [278], Riedmiller und Riedmiller [279], Altenmüller et al. [280], Kuhnle et al. [276], [281]–[283] sowie Dittrich und Fohlmeister [284].

Mahadevan und Theocharous [278] untersuchten schon 1998 DRL zur Steuerung von Transferstraßen, um Methoden der schlanken Produktion zu integrieren und dadurch die Bestände und Liefertreue zu optimieren. Ihr zentrales „Semi-Markov Average Reward Technique“ (SMART)-Modell übertrifft hierbei den Kanban-Benchmark in einem Ein- und einem Drei-Maschinen-Beispiel. Die Produktionsaufträge durchlaufen die Stationen hierfür in fester Sequenz. Es gibt damit kein Auftragsrouting. Ob der Ansatz Auftragseinlastung und Stationsmanagement berücksichtigt, ist nicht ersichtlich. Lediglich Störungen untersuchen die Autoren mit ihrem Ansatz. Der zentrale Agent lernt hierfür Instandhaltungsstrategien, wodurch er die Stationsstörungen reduziert. Die reduzierten Störungen verbessern die Liefertreue im Vergleich zu Kanban. Mit den positiven Ergebnissen stellt dieser Ansatz somit früh eine erfolgreiche DRL-Anwendung im Produktionsbereich dar. Eine direkte Übertragung auf die MP ist aufgrund der nicht eindeutigen Dokumentation sowie der fehlenden Berücksichtigung einiger Entscheidungskategorien der MP nicht möglich.

Ein Jahr später wendeten Riedmiller und Riedmiller in [279] eine Vorstufe von Deep Q-learning zur Lösung eines Produktionssteuerungsproblems in einer Werkstattfertigung an. Dabei ist die Grundidee des Ansatzes dezentraler Natur. In der gegebenen Evaluation testen die Autoren allerdings nur einen Agenten, wodurch die Evaluation als zentral einzustufen ist. Der implementierte Agent repräsentiert eine Arbeitsstation und entscheidet nach Fertigstellung eines Auftrags, welcher als Nächstes in Bearbeitung geht. Damit löst der vorgestellte Ansatz ausschließlich das Stationsmanagement. Im Vergleich zum Steuerungsproblem dieser Arbeit ist die Problemstellung damit weniger komplex, weil nur eine von fünf Entscheidungskategorien zu lösen ist. In ihrer Arbeit zeigen Riedmiller und Riedmiller anhand zwei kleiner Beispiele mit einer und drei Stationen, dass sich DRL im Vergleich zu Standard-Heuristiken besser an lokale Problemstellungen anpasst. Gleichzeitig ermöglicht DRL die Generalisierung auf neue Produktionssituationen. Damit legen Riedmiller und Riedmiller wie Mahadevan und Theocharous [278] einen frühen Grundstein für die DRL-Forschung.

Altenmüller et al. [280] stellen einen aktuellen DRL-Ansatz vor. Sie verwenden einen zen-

tralen DQN-Agenten zur Steuerung eines Produktionsbeispiels aus der Halbleiterindustrie. Die Problemgröße umfasst 10 Stationen, die in fünf verschiedenen Stationsgruppen organisiert sind. Jede Gruppe ist mit einem Puffer ausgestattet. Auftragsrouting und Stationsmanagement entscheidet der Agent, indem er für anfragende Stationen einen Auftrag aus dem Puffer zur Bearbeitung wählt. Auch Störungen berücksichtigt der Ansatz. Der Algorithmus schneidet besser als heuristische Regeln ab, wobei der Fokus auf dem Umgang mit Zeitabhängigkeiten zweier Produktionsschritte liegt. In [280] geben Altenmüller et al. vor allem detaillierte Einblicke in State-, Action- und Reward-Design des DQN-Agenten. Im State integrieren die Autoren u.a. Informationen zur aktuellen Stations- und Pufferbelegung. Außerdem enthält der State die Information, für welche Maschine eine Entscheidung zu treffen ist. Als Actions definieren die Autoren vor allem die Wahl eines spezifischen Auftrags aus dem Puffer für eine Maschine oder die Entscheidung, keinen Auftrag aufzunehmen. Das Reward-Design heben Altenmüller et al. als kritisches Element für erfolgreiches Lernverhalten und zur Erreichung der Produktionsziele hervor. Zusätzlich weisen Altenmüller et al. darauf hin, dass policy-basierte Methoden wie PPO eine konkurrenzfähige Alternative zu value-basierten Methoden wie DQN darstellen, da sie sich als robust gegenüber wechselnden Problemstellungen erweisen. Obwohl der vorgestellte DQN-Agent nicht alle Entscheidungskategorien der MP löst, bietet die Arbeit wertvolle Anhaltspunkte zur Gestaltung eines zentralen DRL-Agenten.

Wie Altenmüller et al. [280] präsentieren Kuhnle et al. [281], [282] einen zentralen DRL-Agenten zur Steuerung eines Produktionsszenarios der Halbleiterindustrie. Im Gegensatz zu Altenmüllers value-basiertem DQN-Algorithmus verwenden Kuhnle et al. die policy-basierte TRPO-Architektur, welche auf Schulman et al. [273] zurückzuführen ist. Der TRPO-Agent zielt auf eine Optimierung von Durchlaufzeiten und Auslastungen in der Halbleiterproduktion ab. Die betrachtete Problemgröße umfasst acht Arbeitsstationen, drei Quellen für Auftragseingänge und ein FTF. Der zentrale TRPO-Agent adressiert alle Entscheidungskategorien der MP zumindest teilweise. Die Auftragseinlastung entscheidet er durch eine Fahrt zu einer Quelle und die Aufnahme eines bereitliegenden Produkts. Stationsmanagement und Auftragsrouting entscheiden sich durch die Wahl der Maschine für ein Produkt. Das Flottenmanagement adressiert der Ansatz zumindest teilweise, weil ein FTF gesteuert wird, welches die Produkte transportiert. Störungen finden ebenfalls Berücksichtigung. Für das Acht-Maschinen-Beispiel übertrifft die Implementierung bestehende Benchmark-Heuristiken. In einer nachfolgenden Arbeit von Kuhnle et al. [283] konzentrieren die Autoren sich auf State-, Action- und Reward-Design der DRL-Produktionssteuerung. Wie Altenmüller et al. folgern sie deren wesentliche Bedeutung für erfolgreiches Lernen. Auch wenn das Halbleiterbeispiel mit lediglich einem Transportfahrzeug arbeitet, lassen sich insbesondere aus [283] Anregungen zu State-, Action- und Reward-Design sowie anderen Setup-Parametern gewinnen. Da alle MP-Entscheidungskategorien zumindest teilweise adressiert werden, bieten die Arbeiten von Kuhnle et al. eine gute Basis zur Implementierung einer DRL-Steuerung für die MP.

Weiterführende Informationen finden sich in Kuhnles Dissertation [276].

Dittrich und Fohlmeister stellen in [284] einen einzelnen DQN-Agenten vor, der mit einem MAS gekoppelt ist, um eine flexible Produktion zu steuern. Aufgrund des zentralen DQN-Moduls ist dieser Ansatz als zentral einzustufen. Das DQN-Modul unterstützt Entscheidungen für anfragende Auftrags-Agenten, die sich zwischen mehreren infrage kommenden Maschinen pro Bearbeitungsschritt entscheiden müssen. Die Entscheidungen betreffen das Auftragsrouting und das Stationsmanagement. Stationsstörungen werden im Artikel erwähnt, aber nicht weiter analysiert. Andere Entscheidungskategorien berücksichtigen die Autoren nicht. Die Simulation der Produktion erfolgt mittels Tecnomatix Plant Simulation. Der Artikel präsentiert Ergebnisse, die für die mittlere Zykluszeit der Produktion zeigen, dass der Ansatz ähnlich gute Ergebnisse liefert wie eine kapazitätsbasierte Planungsheuristik. Ausführliche Untersuchungen bleiben aus. Aufgrund der fehlenden Betrachtung einiger Entscheidungskategorien der MP, der eher knappen Ergebnisanalyse sowie der spezifischen Kopplung von MAS und DQN scheint dieser Ansatz weniger geeignet für das zu untersuchende Steuerungsproblem und findet daher keine weitere Berücksichtigung. Dittrichs und Fohlmeisters Implementierung zeigt jedoch, dass DRL auch in Kombination mit MAS Anwendung finden.

Dezentrale DRL-Ansätze: In dezentralen DRL-Ansätzen sind mehrere lernende Agenten an der Steuerung der Produktion beteiligt. Gabel und Riedmiller [195], [285], Qu et al. [286], Waschneck et al. [277], [287], [288], Bär et al. [289] sowie May et al. [290] steuern die Produktion dezentral lernend.

In den Jahren 2007 und 2008 erarbeiteten Gabel und Riedmiller [195], [285] einen dezentralen DRL-Ansatz zur Steuerung einer Werkstattfertigung als Weiterentwicklung von [279]. Ein DRL-Agent repräsentiert analog zu [279] wiederum eine Station. Jeder Agent wählt nach jeder Auftragsfertigstellung einen neuen zu bearbeitenden Auftrag aus. Die Entscheidungen betreffen somit auch hier ausschließlich das Stationsmanagement, ohne Fokus auf weitere Entscheidungskategorien der MP. Im Vergleich zu [279] enthalten [195], [285] verschiedene Optimierungen zur Steigerung der Dateneffizienz und der Koordination zwischen den Agenten, um das Lernen der unabhängigen Agenten zu fördern. Zusätzlich erfolgt die Evaluation anhand größerer Probleminstanzen, sodass nun bis zu 15 Stationen koordiniert werden, was näher an der Problemgröße dieser Arbeit liegt. Gabel und Riedmiller zeigen erfolgreiches Lernen, vielversprechende Ergebnisse im Vergleich zu Benchmark-Problemen und erklären eine adäquate Generalisierung auf neue Produktionsszenarien. Ergänzende Untersuchungen zeigt Gabel in seiner Dissertation [174]. Insgesamt adressiert der Ansatz mit dem Stationsmanagement lediglich eine von fünf Entscheidungskategorien der MP. Im Gegensatz zu beispielsweise [283] fokussieren sich Gabel und Riedmiller außerdem mehr auf die Koordination zwischen dezentralen Agenten und weniger auf eine umfangreiche Analyse und Dokumentation des Agentendesigns. Letzteres ist für die spätere DRL-Konzeptionierung vorteilhaft.

Daher findet der vorgestellte Ansatz keine weitere Berücksichtigung. Dennoch zeigen Gabel und Riedmiller, dass dezentrale DRL-Ansätze auch global optimale Steuerungsergebnisse erzielen können.

Qu et al. [286] verwenden dezentrales Q-learning zur Steuerung einer Fließfertigung mit parallelen Stationen pro Bearbeitungsschritt, wobei die Approximation der Q-Werte durch ein NN erfolgt. Sie untersuchen eine Problemgröße von sechs Arbeitsstationen, die in drei verschiedene Typen gruppiert sind. Jeder Auftrag durchläuft alle Gruppen sequenziell zur jeweils einmaligen Bearbeitung. Die Besonderheit des Ansatzes ist die Berücksichtigung einer Verteilung von Produktionsarbeitern auf die Stationen. Es handelt sich somit um manuelle Tätigkeiten an Stationen. Ein Agent koordiniert diese Mitarbeiter und führt nach jeder Schicht eine neue Zuweisung durch. Ein zweiter Agent repräsentiert hingegen die Stationen. Er verteilt die im Puffer wartenden Aufträge auf die einzelnen Stationen jeder Gruppe. Letzterer befasst sich also mit den Teilproblemen des Stationsmanagements und Auftragsroutings. Die Auftragseinlastung erfolgt stochastisch verteilt. Die beiden Agenten tauschen Informationen aus, um ein Erreichen des globalen Optimums zu ermöglichen. Ein Rückschluss auf die Performanz des Ansatzes ist allerdings nicht möglich, da die Autoren den Algorithmus nur mittels Parametervariationen untersuchen, nicht aber mit allgemeinen Benchmarks oder theoretischen Optima vergleichen. Außerdem fehlen dem Ansatz Auftragseinlastung, Störungs- und Flottenmanagement. Durch die sequenzielle Bearbeitung in allen Stationsgruppen betrachten Qu et al. zusätzlich keine Mehrfachanfahrten an Stationen, welche im in Abschnitt 2.4.2 beschriebenen Steuerungsproblem dieser Arbeit möglich sind. Der Ansatz zeigt hingegen, dass mittels dezentralem DRL auch andere Teilprobleme einer Produktion wie die Verteilung von Produktionsmitarbeitern gelöst werden können.

Waschneck et al. [287], [288] wenden eine dezentrale DRL-Steuerung auf eine Problemstellung aus der Halbleiterindustrie an. DQN-Agenten steuern Arbeitszentren, die mehrere Maschinen umfassen können. Die Agenten verteilen hierbei abzuarbeitende Produktionslose an die Stationen, womit sie Stationsmanagement und Auftragsrouting lösen. Das Training erfolgt hierfür in verschiedenen Phasen. In einer ersten Phase ist jeweils nur ein Agent aktiv, wobei nacheinander trainiert wird. Prioritätsregeln steuern dabei die jeweils anderen Arbeitszentren. In einer zweiten Phase nutzen alle DQN-Agenten ihr bereits teilweise trainiertes NN. Jedoch trainieren die Agenten wiederum nacheinander. Das heißt, ein Agent ist aktiv und aktualisiert sein NN, die anderen Agenten nutzen das NN nur passiv, um nach der aktuell gelernten Strategie zu agieren. Laut Waschneck et al. beschleunigt und stabilisiert diese Art des Trainings den Trainingsprozess wesentlich. In ihren Simulationen kommen die Autoren zu dem Schluss, dass die Agenten erfolgreich lernen und zufriedenstellende Ergebnisse im Vergleich zu Expertenentscheidungen erzielen. Die Performanz von Benchmark-Heuristiken erreichen die Agenten jedoch nicht. Die Publikationen bieten vor allem wertvolle Anhaltspunkte zur Konzeption einer DRL-Steuerung sowie zur Trainingsgestaltung in dezentralen DRL-Systemen. Einen ergänzenden Überblick gibt [277].

In [289] präsentieren Bär et al. eine dezentrale DQN-Steuerung für ein flexibles Fertigungssystem. Ein Rundförderband verknüpft die Maschinen. Jeder Agent repräsentiert einen Produktionsauftrag. Ein Agent wählt jeweils zwischen zwei Maschinenoptionen für jeden Produktionsschritt aus. Dadurch werden das Auftragsrouting und implizit das Stationsmanagement entschieden. Im Gegensatz zur MP erfolgen die Transporte nicht durch Fahrzeuge, sondern durch das Förderband. Allerdings berücksichtigen die Agenten zumindest Transportzeiten, was einigen anderen Arbeiten fehlt. Das Störungsmanagement zeigen die Autoren nur an einem einfachen Beispiel, bei dem eine Arbeitsstation scheinbar permanent aus der Topologie entfernt wird. Außerdem umfasst das vorgestellte Produktionsbeispiel lediglich sechs Arbeitsstationen sowie drei zu fertigende Produktionsaufträge mit jeweils vier Produktionsschritten. Dadurch werden die Anforderungen an die Entscheidungskategorien, an die Robustheit sowie an die Generalisierbarkeit dieser Arbeit nicht vollständig dargestellt. Allerdings optimieren Bär et al. wie die vorliegende Arbeit den Durchsatz. Im Reward-Design ist hierfür eine Minimierung der Durchlaufzeiten durch die Produktion vorgesehen. Dieses auf die Durchsatzoptimierung ausgerichtete Reward-Design dient aufgrund der positiven Ergebnisse im Vergleich zu einem theoretischen Optimum als Orientierung für die spätere Ansatz-Konzeption.

Schließlich schlagen May et al. [290] eine anpassungsfähige und skalierbare dezentrale DRL-Architektur für eine Matrixproduktion vor, die Auftrags-Agenten, Stations-Agenten und Transport-Agenten einführt. Die Auftrags-Agenten bieten in Auktionen der Stations- und Transport-Agenten, um durch die Produktion zu gelangen. Die Entscheidungen für Gebote können entweder mit Prioritätsregeln oder mit DRL unter Verwendung des PPO-Algorithmus getroffen werden. Auch Stationsentscheidungen werden von PPO-Agenten unterstützt. Jeder Agent verfolgt seine individuellen Ziele. Die dezentrale Steuerung befasst sich folglich mit Auftragsrouting, Stations- und Flottenmanagement. Auch Störungen werden simuliert. Im Vergleich zu [283] arbeiten May et al. mit einer Flotte von 10 Fahrzeugen statt einem einzigen. Die Evaluation von May et al. liegt damit näher an der Problemgröße des hier vorliegenden Steuerungsproblems, wo bis zu 20 Fahrzeuge im Einsatz sind. Die Autoren schlussfolgern, dass dezentrale DRL-Algorithmen in der Lage sind, traditionelle Ansätze in bestimmten Produktionssituationen in einer Matrixproduktion zu übertreffen. Lernen mehrere Agenten gleichzeitig nach ihren individuellen Zielen, leidet jedoch die Systemperformanz. Daher schlagen May et al. in [290] eine hybride Struktur vor, die aus Prioritätsregeln und einigen Lernkomponenten besteht. Weiterhin erweist sich die policy-basierte PPO-Implementierung in Kombination mit dem sogenannten „Action Masking“ als vielversprechend. Action Masking verhindert eine Ausführung nicht valider Actions, sodass sich der Lernprozess auf die validen Actions konzentriert. Dies fördert die DRL-Performanz und Lerngeschwindigkeit. Insgesamt sorgt die dezentrale Problemsicht in diesem Ansatz dafür, dass die Ergebnisse der DRL-Lösungen nicht durchgängig besser sind als Heuristiken. Ein Vergleich zu einer zentralen DRL-Implementierung erfolgt nicht. Eine zentrale Problemsicht

könnte jedoch für die notwendige effiziente Koordination zwischen konkurrierenden Agenten sorgen. In Kombination mit den positiven Ergebnissen anderer oben eingeführter zentraler DRL-Ansätze deutet dies darauf hin, dass eine zentrale Lösung für die MP vorerst zu bevorzugen ist.

5.1.3 Schlussfolgerungen für die Entwicklung der lernenden Steuerung

Tabelle 5.1 analysiert die in Abschnitt 5.1.2 beschriebenen lernenden Steuerungsansätze anhand der Anforderungen der MP (vgl. Abschnitt 2.4.2) an die fünf Entscheidungskategorien Auftragseinlastung, Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement (1.1-1.5) sowie anhand der Anforderungen an die Durchsatzmaximierung bzw. Minimierung der Produktionsspanne (2), die Echtzeitfähigkeit (3), die Datenverfügbarkeit (4) und die Generalisierbarkeit (5) der Ansätze.

Tabelle 5.1: Überblick zu ausgewählten lernenden Steuerungsmethoden in mit der MP verwandten Produktionssystemarten.

Legende: ● berücksichtigt ◐ teilweise berücksichtigt ○ nicht berücksichtigt, nicht ersichtlich	1.1 **Auftragseinlastung**	1.2 **Stationsmanagement**	1.3 **Auftragsrouting**	1.4 **Flottenmanagement**	1.5 **Störungsmanagement**	2 **Durchsatzmaximierung**	3 **Echtzeitfähigkeit**	4 **Datenverfügbarkeit**	5 **Generalisierbarkeit**
Zentrale Ansätze - ein Agent									
Mahadevan und Theocharous (1998)	○	○	○	○	●	○	●	◐	○
Riedmiller und Riedmiller (1999)	○	●	○	○	○	○	●	◐	●
Altenmüller et al. (2020)	◐	●	●	○	●	○	●	◐	○
Kuhnle et al. (2019, 2019b, 2021)	●	●	●	◐	●	◐	●	◐	◐
Dittrich und Fohlmeister (2020)	○	●	●	○	◐	◐	●	◐	◐
Dezentrale Ansätze - mehrere Agenten									
Gabel und Riedmiller (2007, 2008)	○	●	○	○	○	●	●	◐	●
Qu et al. (2016)	○	●	●	○	◐	○	●	◐	◐
Waschneck et al. (2018, 2018b)	◐	●	●	○	◐	○	●	◐	◐
Bär et al. (2020)	○	●	●	◐	◐	●	●	◐	◐
May et al. (2021)	○	●	●	●	●	◐	●	◐	◐

Aufgrund der Eignung für komplexe sequenzielle Entscheidungsprobleme konzentriert sich die vorliegende Arbeit auf die Methode DRL bei der Entwicklung einer lernenden Steue-

rung für die MP. DRL-Ansätze sind dabei entweder zentral oder dezentral organisiert, d.h., entweder ein einzelner Agent oder mehrere Agenten steuern das Produktionssystem. Wie Tabelle 5.1 zeigt, befasst sich die Literatur sowohl mit zentralen als auch mit dezentralen DRL-Ansätzen zur Steuerung von Produktionssystemen. Hierbei lässt sich die Echtzeitfähigkeit aller gelisteten Ansätze für das zu untersuchende Problem als erfüllbar einschätzen. Sie liegt zwar nicht im Fokus der beschriebenen Literatur. Allerdings ermöglicht ein trainiertes NN das Abfragen von Entscheidungen in Echtzeit. Die Trainingsphase ist hierfür zuvor offline durchzuführen. Auch die Datenverfügbarkeit ist kein Kernthema der beschriebenen Literatur. Hingegen liegt der Fokus jeweils auf dem erfolgreichen Training der Agenten, wobei die Lerngeschwindigkeit und nicht die reale Kommunikation von Produktionsentitäten im Vordergrund steht (vgl. z.B. Waschneck et al.). Da DRL-Ansätze entweder zentral oder dezentral agieren, orientiert sich die Bewertung der Datenverfügbarkeit in Tabelle 5.1 an der Literaturanalyse zum zentralen und dezentralen Steuerungsansatz aus den Abschnitten 3.1.3 und 4.1.3, wo die Datenverfügbarkeit bei beiden Paradigmen teilweise gezeigt wird. Deshalb erfolgt die Einschätzung für lernende Steuerungen als „teilweise berücksichtigt“. Alle analysierten Ansätze adressieren zumindest eine der fünf Entscheidungskategorien der MP. Zur Lösung der Entscheidungskategorien wenden die Autoren hierbei value-basierte, policy-basierte sowie Actor-Critic-Methoden an. Am häufigsten werden DQN-Agenten eingesetzt. Bei der Entscheidungsfindung steht auch die Durchsatzmaximierung teilweise im Vordergrund. Arbeiten von Riedmiller und Riedmiller sowie Gabel und Riedmiller legen außerdem Wert auf die Generalisierbarkeit der entwickelten DRL-Ansätze, indem das Agentendesign auf zahlreiche Problemkonfigurationen angewendet wird.

Insgesamt lässt sich feststellen, dass sowohl zentrale als auch dezentrale DRL-Ansätze erfolgreich Anwendung finden. Zu beiden Paradigmen existieren aktuelle Arbeiten. Eine eigene Arbeit in [36] bestätigt für ein MP-Beispiel, dass sich die MP sowohl zentral als auch dezentral mittels DRL steuern lässt. Die Lösungsgüte ergibt sich als gleichwertig.

Von den in der Tabelle analysierten Ansätzen liegen die Arbeiten von Kuhnle et al. [276], [281]–[283] am engsten an den Anforderungen. Kuhnle et al. legen ausführlich dar, wie zentrale DRL-Agenten konzipiert werden können. Ähnlich ausführlich beschreiben Altenmüller et al. [280] die Konzeption der Agenten. Beide legen damit die Basis für erfolgreiches State-, Action- und Reward-Design. Da beide Ansätze zentral organisiert sind und mit einem einzigen DRL-Agenten erfolgreich arbeiten, was sich auch in [36] für die MP bestätigt, scheint die Wahl eines zentralen Agenten als sinnvoll. Jedoch adressieren Kuhnle et al. und Altenmüller et al. jeweils eine Halbleiterproduktion. Deshalb ist eine Adaption auf die MP und eine Erweiterung auf alle Anforderungen des zu untersuchenden Steuerungsproblems erforderlich. Hierbei unterscheidet sich die entwickelte MP-Steuerung im Vergleich zu [283] und [280] vor allem beim Flottenmanagement zur Koordination mehrerer autonomer Fahrzeuge. Eine weitere zu erfüllende Anforderung ist das Ziel der Durchsatzoptimierung. Bär et al. zeigen in [289] eine Gestaltung des Rewards, der eine Durchsatzmaximierung ermöglicht.

Daher orientiert sich der spätere Algorithmus hierfür an dieser Konzeption. Als Grundlage dient die Literatur auch beim sogenannten Action Masking. Hierbei verwenden May et al. [290] Action Masking zur Steigerung der Agentenperformanz. Abschließend ist eine geeignete DRL-Methode zu wählen. Die analysierten Ansätze verwenden überwiegend den DQN-Algorithmus oder Vorläufer hierzu aus dem value-basierten Bereich. Kuhnle et al. setzen stattdessen erfolgreich auf den policy-basierten TRPO-Algorithmus. May et al. verwenden die Actor-Critic-Methode PPO, auf die auch Altenmüller et al. als vielversprechende Alternative zu DQN hinweisen. Insgesamt sind alle genannten DRL-Algorithmen als valide Optionen für die Steuerung der MP zu sehen. Aufgrund der jüngeren Erfolge von PPO sowie der Stabilität und Verlässlichkeit beim Lernen [274] basiert die lernende Steuerung auf der DRL-Methode PPO.

Bevor Abschnitt 5.3 das entwickelte Gesamtkonzept einer lernenden Steuerung erläutert, geht Abschnitt 5.2 auf die DRL-Methode PPO als Kernmethode des Konzepts ein.

5.2 Methode: Deep Reinforcement Learning

DRL verbindet RL mit Deep Learning durch den Einsatz tiefer NNs zur Funktionsapproximation [178, S. 475]. Abschnitt 5.1.3 wählt die DRL-Methode PPO zur Umsetzung der lernenden Steuerung. Für PPO verwenden Schulman et al. [274] ein sogenanntes „Multilayer Perceptron“ (MLP). MLP-Netze lehnen sich an das biologische Vorbild des menschlichen Gehirns an. Die Neuronen werden hierbei oft auch als Perzeptronen bezeichnet. In der Praxis haben MLP-Netze meist mehrere Schichten versteckter Neuronen zwischen einer Input-Schicht und einer Output-Schicht. Dabei ist jedes Perzeptron einer Schicht mit allen Neuronen der Vorgänger- und Nachfolgerschicht verbunden. Anhand des Input-Vektors berechnet sich über die versteckten Schichten hinweg ein Output-Vektor oder -Wert („feedforward“). Die Gewichte der Verbindungen zwischen den Schichten sind zu lernen, um die zu lernende Funktion bestmöglich zu approximieren. Grundlagen hierzu finden sich beispielsweise in [291, S. 4-20], [173, S. 163-220] und [292, S. 77-87]. Im Folgenden verwendet diese Arbeit NN synonym zu MLP. Der Fokus dieses Abschnitts liegt auf den methodischen Grundlagen der DRL-Methode PPO. Hierzu legt Abschnitt 5.2.1 die Basis, indem RL-Grundlagen eingeführt werden. Abschnitt 5.2.2 erläutert anschließend DRL-Grundlagen aus dem policy-basierten DRL-Bereich und verbindet dadurch RL mit Deep Learning. Darauf baut die Actor-Critic-Methode PPO auf, welche Abschnitt 5.2.3 erläutert. Die folgenden Ausführungen vereinheitlichen die variierende Notation der unterschiedlichen Literaturquellen.

5.2.1 Grundlagen zum Reinforcement Learning

Dieser Abschnitt führt in die Methode RL ein. Zuerst wird in Ergänzung zu Abschnitt 2.3.3 die Interaktion des Agenten mit seiner Umgebung formalisiert. Anschließend erfolgt eine

Erfassung der sequenziellen Entscheidungen in der Umgebung als Markov Decision Process (MDP). Es folgen methodische Grundlagen zu Returns und Episoden sowie zur Policy- und Value-Funktion, welche essenziell für das Lernverhalten des Agenten sind. Abschließend stellt der Abschnitt verschiedene Optimierungsmethoden im RL gegenüber. Die Inhalte legen hierbei den notwendigen Grundstein zum Verständnis der Actor-Critic-Methode PPO.

Agent-Umgebungs-Interaktion

RL basiert auf der Interaktion zwischen einem Agenten und seiner Umgebung, die sich gegenseitig beeinflussen. Abbildung 5.1 stellt den Interaktionszyklus der beiden Komponenten nochmals dar (vgl. auch Abschnitt 2.3.3). Der Agent kennt den State $s_t \in \mathcal{S}$ der Umgebung zum jetzigen Zeitpunkt t. Basierend darauf führt er zu diesem Zeitpunkt eine Action $a_t \in \mathcal{A}$ aus. Diese verändert den State der Umgebung im nächsten Zeitschritt $t + 1$ zu s_{t+1}. Die Action führt außerdem zu einem numerischen Reward $r_{t+1} \in \mathcal{R}$ für den Agenten, den er im nächsten Zeitschritt von der Umgebung erhält. Der Reward informiert den Agenten über die Güte der zuvor durchgeführten Action. Das Ziel des Agenten ist es, über andauernde Interaktionen optimale Actions zu erlernen, um die Rewards langfristig zu optimieren. Die Interaktion zwischen Agent und Umgebung lässt sich als Trajektorie τ beschreiben. Startend in State s_0 sieht eine Trajektorie beispielsweise folgendermaßen aus [178, S. 47ff.]:

$$\tau \doteq s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots \tag{5.1}$$

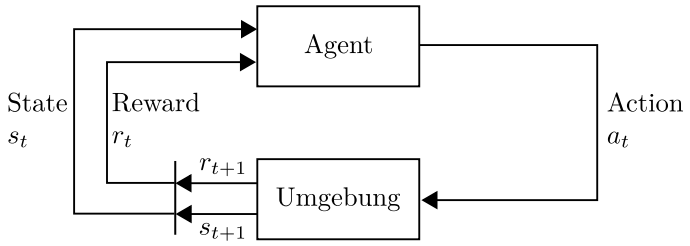


Abbildung 5.1: Agent lernt durch Interaktion mit seiner Umgebung, Quelle: Eigene Darstellung in Anlehnung an Sutton und Barto [178, S. 48].

Markov Decision Process

Um eine mathematische Grundlage zu bilden, kann die oben beschriebene Interaktion zwischen Agent und Umgebung als MDP formuliert werden. Ein MDP beschreibt zeitdiskrete Steuerungsprobleme durch stochastische Transitionen zwischen States basierend auf Actions [179, S. 37]. In einem finiten MDP enthalten $\mathcal{S}$, $\mathcal{A}$ und $\mathcal{R}$ eine endliche Menge von Elementen. In diesem Fall repräsentieren Rewards r_t und States s_t Zufallsvariablen mit diskreten Wahrscheinlichkeitsverteilungen, welche ausschließlich vom vorangegangenen State und der

gewählten Action abhängen [178, S. 48]. Damit lässt sich ein MDP vollständig mit der Wahrscheinlichkeitsverteilung $p(s', r \mid s, a)$ modellieren, welche Transitionen ausgehend von State $s \in \mathcal{S}$ mit Action $a \in \mathcal{A}$ zu State $s' \in \mathcal{S}$ mit Reward $r \in \mathcal{R}$ beschreibt [178, S. 48]:

$$p(s', r \mid s, a) \doteq Pr\{s_t = s', r_t = r \mid s_{t-1} = s, a_{t-1} = a\}. \tag{5.2}$$

Die Funktion p gibt also die Wahrscheinlichkeit an, aus dem State s mit der Action a in den nächsten State s' überzugehen und dabei den Reward r zu erhalten. Die Transitionen im MDP hängen damit ausschließlich vom derzeitigen State s und der gewählten Action a ab, nicht jedoch von den vergangenen. Daher muss der State alle Informationen aus der Vergangenheit der Agent-Umgebungs-Interaktion beinhalten, die für die Zukunftsentscheidungen relevant sind. Dies wird als Markov-Eigenschaft bezeichnet [178, S. 49]. Für die Wahrscheinlichkeitsverteilung $p(s', r \mid s, a)$ gilt:

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{für alle } s \in \mathcal{S},\ a \in \mathcal{A}(s). \tag{5.3}$$

Aus der Funktion $p(s', r \mid s, a)$ lässt sich nun auch eine Transitions-Funktion $p(s' \mid s, a)$, die State-Transitionen beschreibt, ableiten. Diese gibt die Wahrscheinlichkeit für eine Transition von State s zu s' unter Action a an [178, S. 49]:

$$p(s' \mid s, a) \doteq Pr\{s_t = s' \mid s_{t-1} = s, a_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a). \tag{5.4}$$

Zudem beschreibt die Reward-Funktion $r(s, a, s')$ den erwarteten Reward einer gegebenen Transition [178, S. 49]:

$$r(s, a, s') \doteq \mathbb{E}[r_t \mid s_{t-1} = s, a_{t-1} = a, s_t = s'] = \sum_{r \in \mathcal{R}} r \cdot \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}. \tag{5.5}$$

Ist bei einem RL-Problem der zugrundeliegende MDP bekannt, lassen sich zur Optimierung der Actions modellbasierte Methoden anwenden, die diese Informationen nutzen. Andererseits optimieren modellfreie Methoden die Interaktion, ohne den zugrundeliegenden MDP zu kennen [179, S. 204].

Returns und Episoden

Ein MDP kann finale (engl.: terminal) States besitzen, aus denen keine weiteren Transitionen ausgehen und mit denen die Interaktion zwischen Agent und Umgebung endet. Nach Erreichen eines Terminal-States in Zeitschritt T wird das System wieder auf den Initialzustand zurückgesetzt und die Interaktion beginnt von Neuem. In solchen Fällen wird jeder Interaktionsdurchlauf bis zum Terminal-State als Episode bezeichnet. Derartige Problemstellungen

sind als Episodic-Tasks bekannt. Besitzt ein MDP allerdings keine Terminal-States, handelt es sich um eine Continuing-Task. Der Interaktionszyklus bleibt dann unendlich lange bestehen mit Terminal-State bei $T = \infty$ [178, S. 57].

Der Agent hat nun als Ziel, seine kumulativen erwarteten Rewards ab dem aktuellen Zeitpunkt t zu maximieren. Dieses Maximierungsziel wird auch als Return G_t bezeichnet. Bei Episodic-Tasks berechnet sich der Return als Summe aller Rewards ab Zeitpunkt t der Episode [178, S. 54f.]:

$$G_t \doteq r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T. \tag{5.6}$$

Mit dieser Definition wäre G_t bei Continuing-Tasks unendlich, sodass hierbei eine Diskontierung mit dem Discount-Faktor $0 \leq \gamma \leq 1$ erforderlich ist:

$$G_t \doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{5.7}$$

Der Discount-Faktor stellt dabei sicher, dass Rewards aus der ferneren Zukunft exponentiell weniger Einfluss haben. Bei Continuing-Tasks erfolgt damit eine Maximierung des diskontierten Rewards [178, S. 55].

Policy und Value-Funktion

Der Optimierung der Actions liegen im RL die Value-Funktion und die Policy zugrunde. Die Policy $\pi(a \mid s)$ beschreibt das Handeln des Agenten basierend auf dem derzeitigen State s. Sie gibt die Wahrscheinlichkeit wieder, mit der sich der Agent in State s für Action a entscheidet und repräsentiert somit die Strategie des Agenten [178, S. 58]:

$$\pi(a \mid s) \doteq Pr\{a_t = a \mid s_t = s\}. \tag{5.8}$$

Die Value-Funktion $v_\pi(s)$ beschreibt den Wert eines States s, wenn der Agent von s aus der Policy π folgt. Sie wird damit als erwarteter Return mit der Policy π bei Start in State s definiert [178, S. 58]:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid s_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right], \text{für alle } s \in \mathcal{S}. \tag{5.9}$$

Die sogenannte Action-Value-Funktion $q_\pi(s, a)$ hingegen berechnet den erwarteten Return, wenn der Agent in State s die Action a wählt, um dann anschließend der Policy π zu folgen [178, S. 58]:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right]. \tag{5.10}$$

Mit letzterer Action-Value-Funktion lassen sich alternative Actions a aus einem State s bewerten. Je höher der Action-Value-Funktionswert für eine Action, desto lukrativer ist es für den Agenten, diese Action zu wählen. Dabei erwartet der Agent einen höheren Return. Action-Value-Funktionen werden beispielsweise im tabellenbasierten Q-learning gelernt. Sind die Action-Values bekannt, leitet der Agent davon seine Policy ab [293]. Eine DRL-Variante von Q-learning, wobei ein DQN die Action-Value-Funktion approximiert, ist Deep Q-learning [176], [272].

Sowohl $v_\pi(s)$ als auch $q_\pi(s,a)$ hängen von der Policy π ab. Grundsätzlich versucht ein RL-Agent, eine optimale Policy π_* zu finden [178, S. 62]. Hierfür existieren verschiedene Ansätze. Einerseits kann die Value-Funktion berechnet werden, welche dann die optimalen Actions vorgibt. Andererseits kann die optimale Policy π_* auch direkt gelernt werden, ohne den Zwischenschritt über die Value-Funktion. Wie in Abschnitt 5.1.1 bereits beschrieben, konzentrieren sich die folgenden Ausführungen hierfür auf approximative DRL-Methoden, da tabellenbasierte Verfahren aufgrund des Tabellenwachstums mit der State- und Action-Dimension $(\mathcal{S}, \mathcal{A})$ für größere Probleme rechentechnische Grenzen haben.

Policy-basiert, value-basiert und Actor-Critic

Bei komplexen Problemen mit Störfaktoren, beispielsweise in der dynamischen MP, bleibt der zugrundeliegende MDP meist unbekannt oder er ist nicht ausreichend präzise. Dabei führt die Stochastizität durch zufällige Störfaktoren zu vielen verschiedenen potenziell auftretenden Zukunftssituationen bei der Durchführung von Actions [294, S. 49f.]. In der MP treten in diesem Zusammenhang beispielsweise Stationsstörungen unvorhersehbar auf. Insgesamt gilt: Ein Modell der Umgebung, respektive der MDP, ist lediglich hilfreich, wenn es ausreichend genau ist [178, S. 12]. Für den Fall zu ungenauer oder unbekannter Modelle existieren zahlreiche modellfreie DRL-Methoden, um die zugrundeliegenden Dynamiken aus interaktionsbasierten Erfahrungen des Agenten zu erlernen, unabhängig vom Modell. Diese lassen sich, wie bereits in Abschnitt 5.1.1 eingeführt, in value-basiert, policy-basiert und Actor-Critic-Methoden unterteilen, wobei die letztere Herangehensweise die ersten beiden Methoden kombiniert.

Bei value-basierten Methoden wie Q-learning lernt der Agent i.d.R. die Value-Funktion aus (5.9) oder die Action-Value-Funktion aus (5.10) durch die Erfahrungen mittels Interaktion. Daraus ergibt sich implizit die optimale Policy π_* des Agenten, die nach der Höhe der gelernten Values agiert. Ein NN modelliert hierbei die Value-Funktion [178, S. 62ff., S. 198]. Value-basierte Methoden zeichnen sich durch schnelles Lernen aus. Sie benötigen weniger Daten durch Actions, bis eine nahezu optimale Policy gefunden wird. Diese Eigenschaft nennt sich Sample-Effizienz. Gleichzeitig sind die richtigen Hyperparameter bei solchen Methoden häufig schwierig zu finden [295], [296].

Im Gegensatz zur Approximation von Value-Funktionen nutzen policy-basierte Methoden

das NN, um direkt die Policy-Funktion des Agenten zu modellieren und zu optimieren, ohne dabei auf Value-Funktionen zurückzugreifen. Dies ermöglicht das Erlernen komplexerer, stochastischer Policies. Allerdings ist die Auswertung der Policy häufig zeitlich aufwendig, nicht sample-effizient und mit hoher Varianz verbunden [178, S. 321ff.], [295], [296], [274].

Um die Vorteile beider DRL-Arten zu nutzen, verbindet sie die Actor-Critic-Methode. Diese Herangehensweise erlernt sowohl die Value-Funktion als auch die Policy. Das Modell der Policy, auch „Actor“ genannt, interagiert mit der Umgebung und zielt nach wie vor auf die Policy-Optimierung ab. Das Modell der Value-Funktion wird nun aber zusätzlich genutzt, um die Güte der Policy zu evaluieren, weshalb es als „Critic“ bezeichnet wird. Dies erlaubt eine effizientere Policy-Auswertung mit weniger Varianz im Vergleich zu rein policy-basierten Methoden, wobei trotzdem stochastische Policies mit schneller Konvergenz erlernt werden können. Damit gilt diese Herangehensweise in der Literatur derzeit als vielversprechend [295], [274], [179, S. 339-409], [178, S. 331f.].

Wie in Abschnitt 5.1.3 beschrieben, nutzt diese Arbeit PPO als Actor-Critic-Methode. Da Actor-Critic-DRL eine Kombination von value-basiertem und policy-basiertem DRL ist, ist ein Verständnis beider Bereiche nötig. Die folgenden Abschnitte erläutern die notwendigen methodischen Grundlagen. Zuerst werden Policy-Gradient-Methoden aus dem policy-basierten Bereich erklärt, wobei die beiden Begriffe meist synonym verwendet werden, wie beispielsweise in [179, S. 341]. Der anschließende Abschnitt erweitert Policy-Gradient-Methoden hin zu Actor-Critic PPO (AC-PPO).

5.2.2 Policy-Gradient-Methoden

Policy-Gradient-Methoden lernen direkt eine parametrisierte Policy $\pi(a \mid s, \boldsymbol{\theta})$, wobei $\boldsymbol{\theta} \in \mathbb{R}^d$ der Parametervektor ist. Die Policy kann mit verschiedenen Funktionen approximiert werden, solange $\pi(a \mid s, \boldsymbol{\theta}) = Pr\{a_t = a \mid s_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$ bezüglich ihrer Parameter differenzierbar ist. Im DRL wird ein NN zur Approximation der Policy verwendet, wobei der Parametervektor $\boldsymbol{\theta}$ den Gewichten des NNs entspricht [178, S. 321-323], [179, S. 340-349]. Für eine einfachere Notation verwendet diese Arbeit in der Folge:

$$\pi_{\boldsymbol{\theta}}(a \mid s) = \pi(a \mid s, \boldsymbol{\theta}).$$

Das Ziel der Policy-Gradient-Methoden ist es, ein Leistungsmaß $J(\boldsymbol{\theta})$ zu maximieren, indem die NN-Gewichte $\boldsymbol{\theta}$ mittels Gradientenaufstieg gelernt werden. Die formale Beschreibung eines Update-Schritts der Parameter des NNs ist wie folgt [178, S. 321]:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}, \tag{5.11}$$

wobei α die Lernrate und $\widehat{\nabla J(\boldsymbol{\theta}_t)} \in \mathbb{R}^d$ eine stochastische Schätzung ist, dessen Erwartungswert den Gradienten des Leistungsmaßes in Abhängigkeit des Parametervektors $\boldsymbol{\theta}_t$ zum

Zeitpunkt t approximiert.

Das Leistungsmaß für den Fall einer Episodic-Task ist in [178, S. 324] definiert als:

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0), \tag{5.12}$$

wobei $v_{\pi_{\boldsymbol{\theta}}}(s_0)$ die Value-Funktion des Start-States s_0 einer Episode unter der aktuellen Policy $\pi_{\boldsymbol{\theta}}$ ist. Hierbei wird angenommen, dass jede Episode im selben State s_0 startet. Um dieses Leistungsmaß nach und nach zu maximieren, sollte sich die Policy $\pi_{\boldsymbol{\theta}}$ in Richtung von Trajektorien τ (vgl. (5.1)) bewegen, die einen hohen Return liefern, wodurch die Value-Funktion $v_{\pi_{\boldsymbol{\theta}}}(s_0)$ optimiert wird.

Die Wahrscheinlichkeit, dass eine Trajektorie τ durch $\pi_{\boldsymbol{\theta}}$ aus einem definierten State s_0 heraus erzeugt wird, ist [178, S. 104]:

$$p(\tau \mid \pi_{\boldsymbol{\theta}}) \doteq \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t), \tag{5.13}$$

mit $p(s_{t+1} \mid s_t, a_t)$ als Übergangswahrscheinlichkeit von s_t zu s_{t+1} mit der Action a_t (vgl. (5.4)). Dabei bezeichnet $\pi_{\boldsymbol{\theta}}(a_t \mid s_t)$ die Wahrscheinlichkeit, dass a_t in State s_t von der Policy $\pi_{\boldsymbol{\theta}}$ ausgewählt wird (vgl. (5.8)). T ist die Länge der Trajektorie τ bei Erreichen des Terminal-States der Episodic-Task.

Die verallgemeinerte Form für die Wahrscheinlichkeit einer Trajektorie τ aus einem State s_t heraus ist hingegen [178, S. 104]:

$$p(\tau \mid \pi_{\boldsymbol{\theta}}) \doteq \prod_{k=t}^{T-1} \pi_{\boldsymbol{\theta}}(a_k \mid s_k) p(s_{k+1} \mid s_k, a_k). \tag{5.14}$$

Auch hier bezeichnet T die Länge der Trajektorie. (5.15) setzt die Eintrittswahrscheinlichkeiten derselben Trajektorie unter unterschiedlichen Policies $\pi_{\boldsymbol{\theta}'}$ und $\pi_{\boldsymbol{\theta}}$ in ein Verhältnis, was beim sogenannten „Importance-Sampling“ auch für Policy-Gradient-Methoden Anwendung findet [178, S. 104]:

$$p_{t:T-1} \doteq \frac{p(\tau \mid \pi_{\boldsymbol{\theta}'})}{p(\tau \mid \pi_{\boldsymbol{\theta}})} = \frac{\prod_{k=t}^{T-1} \pi_{\boldsymbol{\theta}'}(a_k \mid s_k) p(s_{k+1} \mid s_k, a_k)}{\prod_{k=t}^{T-1} \pi_{\boldsymbol{\theta}}(a_k \mid s_k) p(s_{k+1} \mid s_k, a_k)} = \prod_{k=t}^{T-1} \frac{\pi_{\boldsymbol{\theta}'}(a_k \mid s_k)}{\pi_{\boldsymbol{\theta}}(a_k \mid s_k)}. \tag{5.15}$$

Diese Wahrscheinlichkeitsverhältnisse nutzt AC-PPO später für mehrfache Gradienten-Updates anhand einer Trajektorie, um die Dateneffizienz zu erhöhen. Dabei lassen sich Erfahrungen, die mit einer alten Policy $\pi_{\boldsymbol{\theta}}$ gesammelt wurden, mehrfach verwenden, indem die Returns der Trajektorie in Abhängigkeit der Policies $\pi_{\boldsymbol{\theta}'}$ und $\pi_{\boldsymbol{\theta}}$ gewichtet werden. Eine Erläuterung dessen erfolgt im nächsten Abschnitt bei der Einführung von AC-PPO.

Wie die Gleichungen zeigen, hängt die Leistung einer Trajektorie τ sowohl von der Auswahl

der Actions a_t in States s_t als auch von den Übergangswahrscheinlichkeiten zwischen States s_t und s_{t+1} mit der Action a_t ab. Die State-Übergänge sind hierbei eine Funktion der Umgebung und damit für den Agenten unbekannt. Daher kann der Gradient des Leistungsmaßes, der wiederum für die Optimierung notwendig ist, nicht berechnet werden [178, S. 324-326].

Die Lösung für dieses Problem ist das Policy-Gradient-Theorem. Es liefert eine analytische Lösung zur Ableitung des Leistungsmaßes, ohne dass die Ableitung der State-Verteilung benötigt wird. Nach [178, S. 325, S. 334] besagt das Policy-Gradient-Theorem für eine Episodic- sowie für eine Continuing-Task:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_s \mu(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a \mid s) q_\pi(s, a), \tag{5.16}$$

$$= \mathbb{E}_\pi \left[\sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a \mid s_t) q_\pi(s_t, a) \right], \tag{5.17}$$

wobei $q_\pi(s, a)$ die Action-Value-Funktion und $\pi_{\boldsymbol{\theta}}(a \mid s)$ die Wahrscheinlichkeit der Selektion von Action a in State s ist. Außerdem bezeichnet $\mu(s)$ die Eintrittswahrscheinlichkeit eines States s unter $\pi_{\boldsymbol{\theta}}$. Der Beweis dieses Theorems findet sich beispielsweise in [178, S. 325, S. 334]. Schließlich lässt sich in (5.17) die Ableitung des Leistungsmaßes unter der Annahme, dass der Agent mit Policy $\pi_{\boldsymbol{\theta}}$ eine repräsentative Menge an States durchläuft, wobei er die States s zu gewissen Wahrscheinlichkeiten $\mu(s)$ aufsucht, als empirischer Erwartungswert der Zufallsvariable s_t darstellen [178, S. 326]. Hierzu sammelt der Agent später State-Action-Reward-Tripel in seiner Umgebung.

Ausgehend von (5.17) folgt durch die Erweiterung um $\frac{\pi_{\boldsymbol{\theta}}(a|s_t)}{\pi_{\boldsymbol{\theta}}(a|s_t)}$ Gleichung (5.18). Analog zur Einführung des Erwartungswerts der Zufallsvariable s_t in (5.17) kann dann die Summe der Action-Wahrscheinlichkeiten, unter der Annahme des empirischen Samplings mit Policy $\pi_{\boldsymbol{\theta}}$, durch den Erwartungswert der Zufallsvariable a_t ersetzt werden. Es ergibt sich (5.19) [178, S. 326-328]. Zur Vereinfachung der Schreibweise lässt sich schließlich der Logarithmus der Wahrscheinlichkeit in (5.20) einführen, weil $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t \mid s_t) = \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}}(a_t|s_t)}$ gilt. Nach [297] kann (5.20) weiterhin verallgemeinert werden zu (5.21), wobei $q_\pi(s_t, a_t)$ durch eine allgemeine Funktion Ψ_t ersetzt wird:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\sum_a \pi_{\boldsymbol{\theta}}(a \mid s_t) q_\pi(s_t, a) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a \mid s_t)}{\pi_{\boldsymbol{\theta}}(a \mid s_t)} \right], \tag{5.18}$$

$$= \mathbb{E}_\pi \left[q_\pi(s_t, a_t) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a_t \mid s_t)}{\pi_{\boldsymbol{\theta}}(a_t \mid s_t)} \right], \tag{5.19}$$

$$= \mathbb{E}_\pi \left[q_\pi(s_t, a_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t \mid s_t) \right], \tag{5.20}$$

$$= \mathbb{E}_\pi \left[\Psi_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t \mid s_t) \right]. \tag{5.21}$$

Je nach gewähltem Policy-Gradient-Algorithmus steht Ψ_t für eine Funktion, die wie die Action-Value-Funktion $q_\pi(s_t, a_t)$ die Güte von Actions aus States heraus bewertet. Wird Ψ_t beispielsweise mit dem Return G_t gleichgesetzt, ergibt sich der sogenannte REINFORCE-Algorithmus, den Williams [298] einführte.

Damit ist die stochastische Schätzung des Policy-Gradienten aus (5.11) zu (5.21) bestimmt, was als Grundlage für den policy-basierten Teil von AC-PPO dient.

5.2.3 Actor-Critic Proximal Policy Optimization

Policy-Gradient-Methoden approximieren lediglich die Policy-Funktion. Im Gegensatz dazu approximieren Actor-Critic-Methoden sowohl die Policy- als auch die Value-Funktion [294, S. 40]. Die Policy wird als Actor bezeichnet, da sie für die Wahl der Actions, d.h. für das Interagieren mit der Umgebung, verantwortlich ist. Die geschätzte Value-Funktion gilt als Critic, da sie die Güte der Handlungen des Actors bewertet. Diese Bewertung kommt in Form eines sogenannten „Temporal Difference“(TD)-Fehlers. Er stellt den einzigen Output des Critics dar und treibt das Lernen für beide Entitäten an [299, S. 151]. Die Interaktion zwischen Actor und Critic stellt Abbildung 5.2 dar. Der Actor sieht im Interaktionszyklus zwei Inputs: den State und den TD-Fehler, der sozusagen das Reward-Signal für den Actor repräsentiert. Auf Basis dieser Informationen lernt er die Policy und wählt die Actions, die in der Umgebung ausgeführt werden. Der Critic empfängt ebenfalls den State als Input. Außerdem erhält er das direkt von der Umgebung zurückgegebene Reward-Signal. Basierend auf State und Reward berechnet der Critic den TD-Fehler, aktualisiert damit seine Value-Funktion und sendet den TD-Fehler außerdem als Feedback an den Actor [299, S. 151-153].

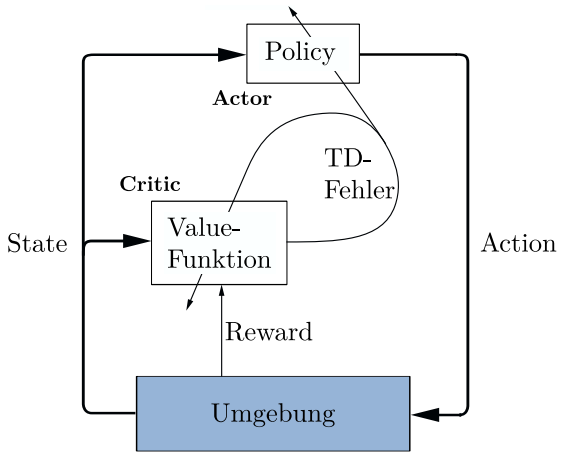


Abbildung 5.2: Die Actor-Critic-Architektur, Quelle: Eigene Darstellung in Anlehnung an [299, S. 151].

Der TD-Fehler δ_t auf Basis der Value-Funktion in Zeitschritt t ist [299, S. 152]:

$$\delta_t = r_t + \gamma v_\pi(s_{t+1}) - v_\pi(s_t), \tag{5.22}$$

wobei v_π die Value-Funktion ist und r_t der zum Zeitschritt t erhaltene Reward. Die Variable γ repräsentiert den angewendeten Discount-Faktor und bestimmt, wie weit der Agent zur Bewertung von States in die Zukunft schaut. Der TD-Fehler kommt hierbei aus dem Bereich des TD-Lernens, nachzulesen in [178, S. 119-140]. Der TD-Fehler vergleicht den geschätzten Value $v_\pi(s_t)$ von State s_t mit der aktuellen Erfahrung $r_t + \gamma v_\pi(s_{t+1})$, die der Agent mit dem Reward r_t gemacht hat. In diesem Umstand begründet sich auch der Name „Temporal Difference“, weil die Values zweier unterschiedlicher bzw. aufeinanderfolgender Zeitpunkte verglichen werden. Für jeden Zeitschritt t in der Umgebung kann ein Update im NN erfolgen [178, S. 119-121]. Ist der TD-Fehler positiv, sollte die Tendenz a_t zu wählen, gestärkt werden und umgekehrt [299, S. 152].

Die Policy- und Value-Funktion einer Actor-Critic-Methode können entweder mit zwei separaten NNs oder mit einem gemeinsamen NN approximiert werden, wobei die Parameter der Schichten für letzteren Fall gemeinsam genutzt werden und für beide Funktionen je ein Output existiert. Der Output der Policy-Funktion hat hierbei die Größe der möglichen Actions $\mathcal{A}$. Der Output der Value-Funktion gibt lediglich den Wert $v_\pi(s_t)$ aus, worauf basierend der TD-Fehler zu berechnen ist [179, S. 351, S. 365].

Auch AC-PPO kann sowohl mit geteiltem NN als auch mit separaten Netzen implementiert werden. Hierbei nutzt AC-PPO die in Abbildung 5.2 dargestellte Actor-Critic-Architektur und geht aus den Arbeiten von Schulman et al. [274], [297] hervor. Die folgenden Ausführungen basieren im Wesentlichen auf diesen beiden Arbeiten und gehen zuerst darauf ein, wie Ψ_t aus (5.21) in AC-PPO umgesetzt ist. Hier wird „Generalized Advantage Estimation“ (GAE) relevant. Danach gehen die Abschnitte auf das Lernen der Policy- (Actor) sowie das Lernen der Value-Funktion (Critic) ein. Eine Zusammenfassung von AC-PPO schließt den Abschnitt ab. Die verfügbaren Quellen zu AC-PPO beschreiben jeweils nur einzelne Teilbereiche des Algorithmus. Online verfügbare Programmcode-Repositories sowie die OpenAI-Dokumentation komplettieren die in [274], [297] dokumentierten Grundlagen deshalb für die folgenden Ausführungen.

Generalized Advantage Estimation

In AC-PPO ersetzt der sogenannte „Advantage-Estimate“ $A_t^{\mathrm{GAE}(\gamma,\lambda)}$ das Ψ_t aus (5.21), was zu folgendem Gradienten führt:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi\Big[A_t^{\mathrm{GAE}(\gamma,\lambda)} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t \mid s_t)\Big]. \tag{5.23}$$

$A_t^{\mathrm{GAE}(\gamma,\lambda)}$ ist hierbei nicht zu verwechseln mit einer Action a_t eines Zeitschritts. GAE wurde erstmals in [297] eingeführt, um die Varianz bei der Schätzung der Policy-Gradienten zu reduzieren sowie die Stabilität beim Lernen zu verbessern. Die Reduktion der Varianz[1]

[1] Varianz entsteht durch zu geringe statistische Robustheit der gelernten Modellparameter [291, S. 174].

geht hierbei auf Kosten des sogenannten Bias[2] im NN. Im Machine Learning ist das Bias-Varianz-Dilemma ein häufiges Problem, bei dem beide Größen minimiert werden sollten, um Generalisierungsfähigkeit sicherzustellen. Die Verringerung der einen Größe erhöht jedoch die andere Größe und umgekehrt. Näheres hierzu findet sich beispielsweise in [291, S. 174-177].

Der Advantage-Estimate $A_t^{\mathrm{GAE}(\gamma,\lambda)}$, auch bekannt als Advantage-Funktion, adressiert dieses Dilemma und ist hierfür in [297] wie folgt definiert:

$$A_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \tag{5.24}$$

wobei γ und λ Discount-Faktoren sind, und δ_{t+l} der TD-Fehler ist, der gemäß (5.22) berechnet wird. Den Input zur Berechnung des TD-Fehlers liefert hierbei die Value-Funktion des Critics (vgl. Abbildung 5.2). Die beiden Discount-Faktoren γ und λ adressieren das Bias-Varianz-Dilemma, übernehmen aber unterschiedliche Rollen. Hierzu lassen sich zwei Spezialfälle betrachten, $\lambda = 0$ und $\lambda = 1$:

$$\mathrm{GAE}(\gamma, 0): \quad A_t^{\mathrm{GAE}(\gamma,0)} = \delta_t \qquad = r_t + \gamma v_\pi(s_{t+1}) - v_\pi(s_t), \tag{5.25}$$

$$\mathrm{GAE}(\gamma, 1): \quad A_t^{\mathrm{GAE}(\gamma,1)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - v_\pi(s_t). \tag{5.26}$$

Im Fall $\lambda = 0$ reduziert sich die Advantage-Funktion zu (5.25) in Äquivalenz zum TD-Fehler aus (5.22). Im Fall $\lambda = 1$ reduziert sich die Advantage-Funktion zu (5.26). $\mathrm{GAE}(\gamma, 1)$ hat hohe Varianz wegen der Aufsummierung der Rewards. $\mathrm{GAE}(\gamma, 0)$ hat hingegen einen höheren Bias aufgrund der Value-Schätzungen. Die Einführung von $0 < \lambda < 1$ erlaubt es, einen Kompromiss zu finden.

Unabhängig von λ reguliert γ die Skala der Value-Funktion v_π. Der Parameter bestimmt, wie viel zukünftig erwartete Rewards zur Schätzung von $A_t^{\mathrm{GAE}(\gamma,\lambda)}$ beitragen sollen (analog zu (5.7)). Je näher γ an 1 liegt, desto kleiner ist der Bias, weil dem Agenten durch den weiteren Blick in die Zukunft mehr Informationen zur Verfügung stehen. Steigt bei geringerem Bias die Varianz, greift die Regulierung von λ ein. Schulman et al. [297] fanden außerdem empirisch heraus, dass $\lambda < 1$ einen weitaus geringeren Bias einführt als γ. Die optimale $\gamma - \lambda$-Parameterkombination ist vom Anwendungsfall abhängig.

Zusammenfassend bewertet die Advantage-Funktion $A_t^{\mathrm{GAE}(\gamma,\lambda)}$ die Güte einer Action und gibt darauf basierend an, ob $\pi_{\boldsymbol{\theta}}(a_t \mid s_t)$ zu erhöhen oder zu verringern ist. Auf Basis des TD-Fehlers, welcher aus der Value-Funktion des Critics zu berechnen ist, stellt die Advantage-Funktion hierbei das Feedback zum Lernen des Actors bereit (vgl. Abbildung 5.2). Wie der Actor die Advantage-Funktion beim NN-Update nutzt, zeigen die folgenden Ausführungen.

[2] Bias bezeichnet den von vereinfachenden Modellannahmen verursachten Modellfehler [291, S. 174].

Actor: PPO-spezifisches Gradienten-Update

AC-PPO zeichnet sich im Wesentlichen durch zwei Merkmale aus:

- **Stabilität durch parallele Trajektorien**: Das Sammeln einzelner Trajektorien sorgt für ein Rauschen in den Daten, da eine einzige Trajektorie aufgrund der Stochastizität der Policy nicht repräsentativ ist. Das synchronisierte Sammeln von n_{env} parallelen Trajektorien adressiert dieses Problem. Der Agent lernt über die Gradienten der aggregierten Erfahrungen durch die parallelen Trajektorien [179, S. 398].

- **Dateneffizienz durch Mehrfach-Updates**: Im Vergleich zu DQN-Agenten gehen die Daten gesammelter Trajektorien bei AC-PPO direkt nach dem NN-Update verloren. Sie können im späteren Lernverlauf nicht wiederholt genutzt werden. Deshalb verwendet AC-PPO die parallel gesammelten Trajektorien direkt für mehrfache NN-Updates, bevor der Agent Folgetrajektorien sammelt. Hierfür nutzt AC-PPO das Konzept des sogenannten „Importance-Samplings“ (vgl. z.B. [178, S. 104]) [179, S. 398].

Das Importance-Sampling aus (5.15) ermöglicht Mehrfach-Updates (Anzahl n_{epochs}) durch Wiederverwendung der Daten einer gesammelten Trajektorie. Die State-Action-Reward-Tripel sind hierzu nach [274] anhand ihrer Eintrittswahrscheinlichkeiten bei einer neuen Policy $\pi_{\boldsymbol{\theta}'}$ im Vergleich zu den Eintrittswahrscheinlichkeiten bei einer alten Policy $\pi_{\boldsymbol{\theta}}$ zu bewerten, indem das Wahrscheinlichkeitsverhältnis $R_t(\boldsymbol{\theta}', \boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}'}(a_t|s_t)}{\pi_{\boldsymbol{\theta}}(a_t|s_t)}$ mit dem Gradienten des Leistungsmaßes in (5.23) multipliziert wird[3]:

$$\nabla_{\boldsymbol{\theta}'} J(\boldsymbol{\theta}', \boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[\frac{\pi_{\boldsymbol{\theta}'}(a_t \mid s_t)}{\pi_{\boldsymbol{\theta}}(a_t \mid s_t)} \cdot A_t^{\text{GAE}(\gamma,\lambda)} \frac{\nabla_{\boldsymbol{\theta}'} \pi_{\boldsymbol{\theta}'}(a_t \mid s_t)}{\pi_{\boldsymbol{\theta}'}(a_t \mid s_t)}\right], \tag{5.27}$$

wobei im Vergleich zu (5.23) $\nabla_{\boldsymbol{\theta}'} \log \pi_{\boldsymbol{\theta}'}(a_t \mid s_t) = \frac{\nabla_{\boldsymbol{\theta}'} \pi_{\boldsymbol{\theta}'}(a_t|s_t)}{\pi_{\boldsymbol{\theta}'}(a_t|s_t)}$ gilt. Die einmal gesammelten State-Action-Reward-Tripel der Trajektorie lassen sich damit unter minimal unterschiedlichen Eintrittswahrscheinlichkeiten mehrfach nutzen, sozusagen recyceln. (5.27) lässt sich umformen zu:

$$\nabla_{\boldsymbol{\theta}'} L_{sur}(\boldsymbol{\theta}', \boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[A_t^{\text{GAE}(\gamma,\lambda)} \frac{\nabla_{\boldsymbol{\theta}'} \pi_{\boldsymbol{\theta}'}(a_t \mid s_t)}{\pi_{\boldsymbol{\theta}}(a_t \mid s_t)}\right], \tag{5.28}$$

wobei L_{sur} als „Surrogate-Objective“ bezeichnet wird. Wie $J(\boldsymbol{\theta})$ ist das Surrogate-Objective zu maximieren. Abbildung 5.3 zeigt die Funktionalität der Mehrfach-Updates.

Im ersten Update-Schritt existiert dabei noch keine neue Policy $\pi_{\boldsymbol{\theta}'}$. Deshalb gilt für diesen Fall: $R_t(\boldsymbol{\theta}' = \boldsymbol{\theta}, \boldsymbol{\theta}) = 1$. $\boldsymbol{\theta}'$ ist im ersten Update-Schritt damit immer äquivalent zu $\boldsymbol{\theta}$, was es in der Folge bei den Begriffen „neue“ und „alte“ Policy zu beachten gilt. Durch jeden Update-Schritt ergibt sich dann eine neue Policy ($\boldsymbol{\theta}'$, $\boldsymbol{\theta}''$, $\boldsymbol{\theta}'''$). Zur Vereinfachung bezeichnet $\boldsymbol{\theta}'$ in den folgenden Ausführungen durchgängig den Parametervektor, der sich durch die NN-Updates

[3] R_t ist hierbei nicht zu verwechseln mit einem Reward r_t eines Zeitschritts.

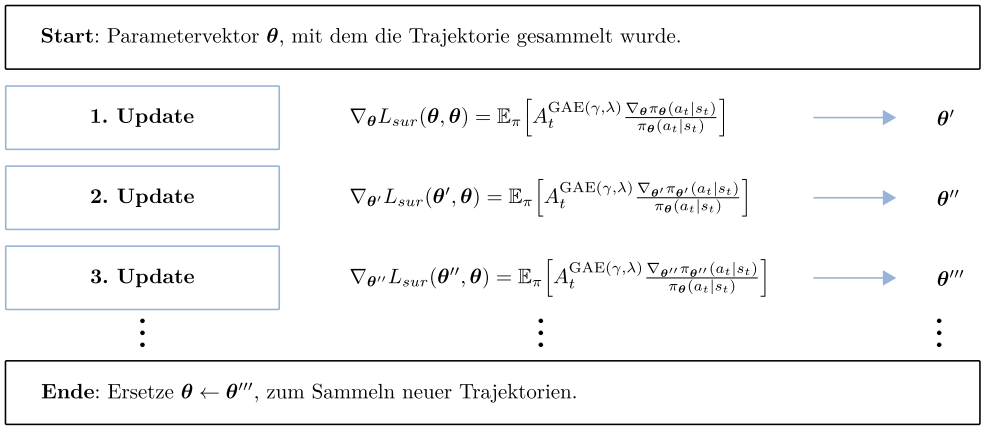


Abbildung 5.3: Darstellung der Funktionalität von Mehrfach-Updates.

ändert. $\boldsymbol{\theta}$ findet hingegen ausschließlich Anwendung, wenn über alle Update-Schritte hinweg die NN-Gewichte vor dem ersten Update-Schritt gemeint sind.

Damit bei den Mehrfach-Updates keine Stabilitätsprobleme durch zu optimistische Schätzungen auftreten, führen Schulman et al. [274] in (5.29) eine „Clip-Funktion“ ein. Diese beschränkt $R_t(\boldsymbol{\theta}', \boldsymbol{\theta})$ durch eine Ober- und eine Untergrenze, als Restriktion für die Policy-Veränderungen. Hierzu dient ein Hyperparameter ϵ, der $R_t(\boldsymbol{\theta}', \boldsymbol{\theta})$ auf das Intervall $[1-\epsilon, 1+\epsilon]$ begrenzt. Gleichzeitig akzeptiert die neue Funktion jedoch pessimistische Änderungen, indem das Minimum aus der unbeschränkten und beschränkten Funktion berechnet wird. L_{sur} wird dadurch zu L_{sur}^{clip} und das Leistungsmaß bestimmt sich zu:

$$L_{sur}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta}) = \mathbb{E}_{\pi}\Big[\min(R_t(\boldsymbol{\theta}', \boldsymbol{\theta}) \cdot A_t^{\mathrm{GAE}(\gamma,\lambda)},\ \mathrm{clip}(R_t(\boldsymbol{\theta}', \boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) \cdot A_t^{\mathrm{GAE}(\gamma,\lambda)})\Big]. \quad (5.29)$$

Abbildung 5.4 zeigt $L_{sur}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta})$ in Abhängigkeit von $R_t(\boldsymbol{\theta}', \boldsymbol{\theta})$ für einen positiven Advantage (a) und einen negativen Advantage (b). Bei positivem Advantage ($A_t^{\mathrm{GAE}(\gamma,\lambda)} > 0$) erhöht der erste Update-Schritt die Wahrscheinlichkeit der jeweiligen Action a_t, was im zweiten Update zu $R_t(\boldsymbol{\theta}', \boldsymbol{\theta}) > 1$ führt. Die Clip-Funktion beschränkt das Wahrscheinlichkeitsverhältnis dann auf eine Obergrenze $1+\epsilon$, um eine zu optimistische Schätzung zu vermeiden. Im Gegensatz dazu führt ein negativer Advantage ($A_t^{\mathrm{GAE}(\gamma,\lambda)} < 0$) zu einem verminderten Gewicht $R_t(\boldsymbol{\theta}', \boldsymbol{\theta}) < 1$. Um zu verhindern, dass die jeweiligen Aktualisierungen der Policy dabei zu klein werden, bildet die Clip-Funktion hier eine Untergrenze $1-\epsilon$ für $R_t(\boldsymbol{\theta}', \boldsymbol{\theta})$, sodass pessimistische Änderungen zulässig sind.

Schließlich ergibt sich das Gradienten-Update in der Schreibweise für die Mehrfach-Updates mit einer neuen und einer alten Policy des Actors in AC-PPO zu:

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}' + \alpha \nabla_{\boldsymbol{\theta}'} L_{sur}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta}). \quad (5.30)$$

In die Gradientenberechnung fließt beim Actor die Advantage-Funktion mit ein, welche den

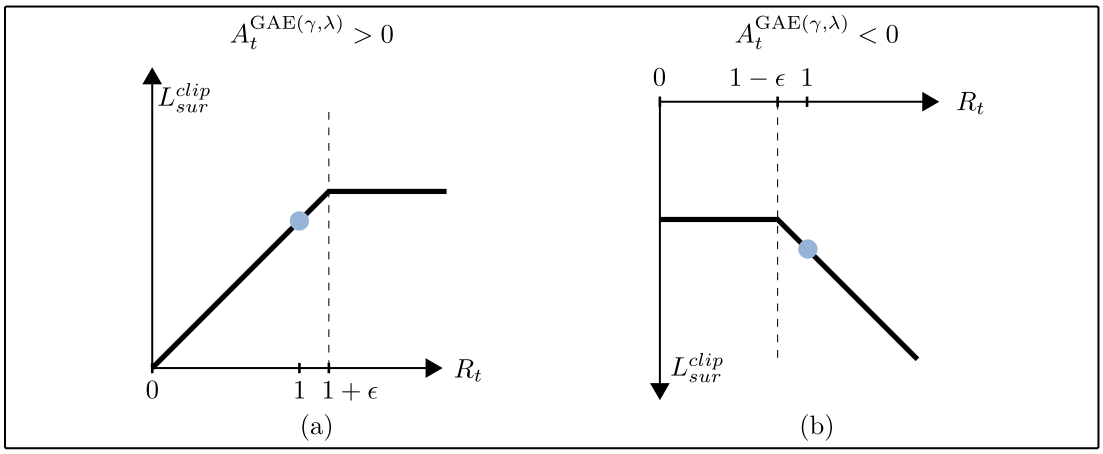


Abbildung 5.4: Darstellung des Leistungsmaßes L_{sur}^{clip} bei $A_t^{\mathrm{GAE}(\gamma,\lambda)} > 0$ (a) und $A_t^{\mathrm{GAE}(\gamma,\lambda)} < 0$ (b), Quelle: Eigene Darstellung in Anlehnung an [274].

TD-Fehler berücksichtigt. Die Berechnung des TD-Fehlers basiert wiederum auf der Value-Funktion des Critics. Die folgenden Ausführungen beschreiben deshalb die Approximation der Value-Funktion aufseiten des Critics.

Critic: Approximation der Value-Funktion

Actor und Critic nutzen in diesem Konzept ein geteiltes NN, was Rechenzeit spart. Diese Konfiguration erwies sich in der vorliegenden Arbeit als erfolgreich. Ein Vergleich zu einer Architektur mit separaten Netzen in Voruntersuchungen ergab keine Nachteile. Die Grundlagen für die Anwendung eines geteilten NNs legen Schulman et al. in [274] dar. Damit teilen sich Actor und Critic auch den Parametervektor $\boldsymbol{\theta}$. Der Critic optimiert die Value-Funktion $v_{\pi_{\boldsymbol{\theta}}}(s_t)$ der States s_t (vgl. Abbildung 5.2), indem er bei AC-PPO die mittlere quadratische Abweichung (MQA) zwischen den vorhergesagten Werten $v_{\pi_{\boldsymbol{\theta}'}}(s_t)$ und den wahren Zielwerten $\hat{v}(s_t)$ aller Samples t aller n_{env} parallel gesammelter Trajektorien τ mit der jeweiligen Länge n_τ minimiert:

$$\min_{\boldsymbol{\theta}'} \quad \frac{1}{n_{\mathrm{env}} \cdot n_\tau} \cdot \sum_\tau \sum_t (v_{\pi_{\boldsymbol{\theta}'}}(s_t) - \hat{v}(s_t))^2. \tag{5.31}$$

$\boldsymbol{\theta}'$ stellt den aktualisierten Parametervektor im NN dar, da auch hier mehrfache Update-Schritte in Bezug auf die Zielwerte $\hat{v}(s_t)$ erfolgen. Da die wahren Zielwerte $\hat{v}(s_t)$ nicht bekannt sind, sind sie vor jedem ersten Update-Schritt zu schätzen. In [179, S. 363] ist das geschätzte Value-Target $\hat{v}(s_t)$ definiert als:

$$\hat{v}(s_t) = v_{\pi_{\boldsymbol{\theta}}}(s_t) + A_t^{\mathrm{GAE}(\gamma,\lambda)}, \tag{5.32}$$

wobei $v_{\pi_{\boldsymbol{\theta}}}(s_t)$ die approximierte Value-Funktion des alten Parametervektors $\boldsymbol{\theta}$ und $A_t^{\mathrm{GAE}(\gamma,\lambda)}$ die approximierte Advantage-Funktion mit $\boldsymbol{\theta}$ ist. Durch Einsetzen von (5.32) in (5.31) ergibt

sich die zu minimierende Funktion zu:

$$L^{\mathrm{VF}}(\boldsymbol{\theta}', \boldsymbol{\theta}) = \frac{1}{n_{\mathrm{env}} \cdot n_\tau} \cdot \sum_\tau \sum_t (v_{\pi_{\boldsymbol{\theta}'}}(s_t) - v_{\pi_{\boldsymbol{\theta}}}(s_t) - A_t^{\mathrm{GAE}(\gamma,\lambda)})^2. \tag{5.33}$$

Das Gradienten-Update in der Schreibweise für die Mehrfach-Updates mit der Lernrate α ist für die MQA-Minimierung des Critics mittels Gradientenabstieg definiert durch:

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}' - \alpha \nabla_{\boldsymbol{\theta}'} L^{\mathrm{VF}}(\boldsymbol{\theta}', \boldsymbol{\theta}). \tag{5.34}$$

Kombination von Actor und Critic

Um Actor und Critic im geteilten NN zu aktualisieren, sind die Zielfunktionen nach [274] zu kombinieren. (5.35) setzt hierbei die Gleichungen (5.29) und (5.33) zu einer gemeinsamen Optimierungsfunktion für die gesammelten Trajektorien zusammen. $L_{sur}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta})$ ist hierbei auf alle Samples t aller n_{env} parallel gesammelter Trajektorien τ mit der jeweiligen Länge n_τ anzuwenden und wird zu: $L_{sur,\tau}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta}) = \frac{1}{n_{\mathrm{env}} \cdot n_\tau} \cdot \sum_\tau \sum_t L_{sur}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta})$ [4]. Schulman et al. [274] fügen zusätzlich einen Entropie-Term $S(\boldsymbol{\theta}')$ als Funktion des Parametervektors hinzu, um die Exploration des Algorithmus zu fördern sowie lokale Optima zu vermeiden [298], [300]. Die Berechnung der über alle Trajektorien gemittelten Entropie erfolgt dabei nach Shannon [301, S. 393ff.]. Die kombinierte Optimierungsfunktion lässt sich ausdrücken als:

$$L(\boldsymbol{\theta}', \boldsymbol{\theta}) = L_{sur,\tau}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta}) - c_1 \cdot L^{\mathrm{VF}}(\boldsymbol{\theta}', \boldsymbol{\theta}) + c_2 \cdot S(\boldsymbol{\theta}'), \tag{5.35}$$

wobei c_1 der sogenannte Value-Funktions-Koeffizient ist und Werte zwischen 0.5 und 1 annimmt. c_1 gewichtet die Terme von Actor sowie Critic und bringt sie in ein passendes Verhältnis. Durch einen Vorzeichenwechsel wird hierbei die MQA-Minimierung als Maximierung dargestellt. c_2 steuert hingegen den Einfluss des Entropie-Terms und damit die Exploration durch leichte Variationen im Leistungsmaß. Das gesamte Gradienten-Update erfolgt dann via Gradientenaufstieg mit der Lernrate α zu:

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}' + \alpha \nabla_{\boldsymbol{\theta}'} L(\boldsymbol{\theta}', \boldsymbol{\theta}). \tag{5.36}$$

Zusammenfassend ist der Pseudocode für AC-PPO in Alg. 5.1 dargestellt. Der Algorithmus beginnt mit der Initialisierung des NN-Parameters $\boldsymbol{\theta}$ sowie der weiteren Hyperparameter, wobei α die Lernrate des geteilten NNs ist. Dann werden n_{env} Trajektorien der Länge n_τ erzeugt, indem der Agent der aktuellen Policy folgt. Es folgt die Berechnung der Advantage-Estimates sowie der Value-Targets für jede Trajektorie. Schließlich nutzt AC-PPO diese Werte für n_{epochs} Updates, wobei der Parametervektor $\boldsymbol{\theta}$ über Gradientenaufstieg mittels

[4] Hierbei berechnet sich $L_{sur}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta})$ für jeweils ein State-Action-Reward-Tripel. Die Darstellung als Erwartungswert aus (5.29) ist dabei zu vernachlässigen. Der Erwartungswert wird mit dem Durchschnitt der gesammelten Daten approximiert.

(5.36) zu aktualisieren ist. Dieser Prozess wiederholt sich in der Lernphase des Agenten für eine vorgegebene Anzahl an n_{episodes} Episoden.

Algorithmus 5.1 AC-PPO in Anlehnung an [274], [297].

1: Initialisiere Hyperparameter: $n_{\text{episodes}}, n_{\text{epochs}}, n_{\text{env}}, n_\tau, \epsilon, \alpha, c_1, c_2$ etc.
2: Initialisiere den Parametervektor $\boldsymbol{\theta} \in \mathbb{R}^d$.
3: Initialisiere n_{env} parallele Umgebungen.
4: **for** $i \leftarrow 1$ to n_{episodes} **do**
5: Generiere n_{env} Trajektorien der Länge n_τ unter Policy $\pi_{\boldsymbol{\theta}}(\cdot \mid \cdot)$.
6: Speichere die Action-Wahrscheinlichkeiten als $\pi_{\boldsymbol{\theta}}(a \mid s)$.
7: Berechne die Advantage-Estimates $A_t^{\text{GAE}(\gamma,\lambda)}$ via (5.24).
8: Berechne die Value-Targets $\hat{v}$ via (5.32).
9: **for** $j \leftarrow 1$ to n_{epochs} **do**
10: Aktualisiere $\boldsymbol{\theta}$ mittels Gradientenaufstieg: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}' + \alpha \nabla_{\boldsymbol{\theta}'} L(\boldsymbol{\theta}', \boldsymbol{\theta})$.
11: **end for**
12: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$
13: **end for**

5.3 Konzeptionierung des Deep Reinforcement Learning-Agenten

Abschnitt 5.1.3 leitet aus dem bestehenden Stand der Forschung ein Grobkonzept zur lernenden DRL-Steuerung der MP ab, vor allem auf Basis von [276], [281]–[283]. Abschnitt 5.2 stellt anschließend AC-PPO als gewählte DRL-Methode vor, welche die zentrale Koordination der MP lernt und umsetzt. Darauf basierend fasst Abbildung 5.5 das auf das Steuerungsproblem dieser Arbeit angepasste Feinkonzept des lernenden Steuerungsansatzes zusammen und deutet überblicksweise die Lösung der fünf Entscheidungskategorien aus Abschnitt 2.2.2 an. Teile dieses Ansatzes wurden in [34]–[36] veröffentlicht.

Der zentrale PPO-Agent lernt und koordiniert die fünf Entscheidungskategorien der MP durch direkte Koordination der Fahrzeuge. In der Lernphase interagiert der Agent mit den Fahrzeugen in der Umgebung (s_t, r_t, a_t), sammelt dadurch Erfahrungen und speichert diese Erfahrungen im NN (nach Alg. 5.1). In der Steuerungsphase führt er das Gelernte aus. In beiden Fällen wird der Agent aktiv, wenn ein Fahrzeug einen Fahrauftrag benötigt: d.h. im Initialzustand der MP, wenn noch keine Fahraufträge vergeben wurden, nach der Aufnahme eines Auftrags in der Quelle, bei Fertigstellung eines Auftrags in der Senke und wenn das ihm zugewiesene Produkt an eine neue Station transportiert werden muss. Die Auftragseinlastung steuert der Agent, indem er ein anfragendes Fahrzeug ohne zugewiesenes Produkt zur Quelle schickt und ihm einen zu startenden Produkttyp zuweist, in Anlehnung an [283]. Ist einem Fahrzeug einmal ein Produkt zugewiesen, begleitet es den Auftrag nach

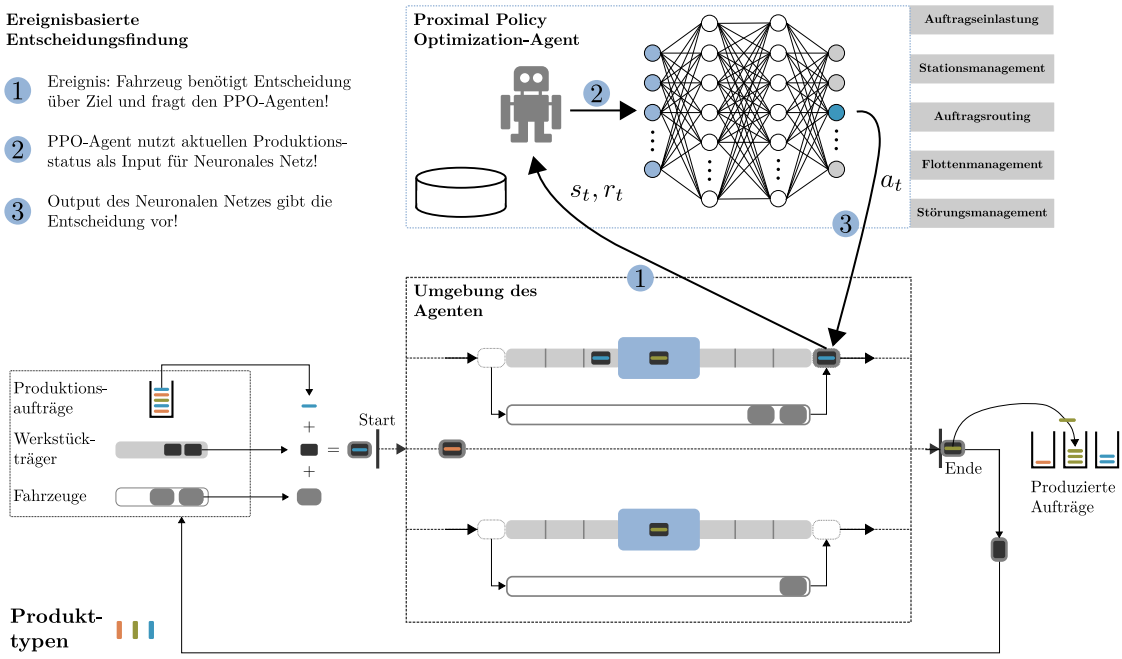


Abbildung 5.5: Lernender PPO-Agent zur Steuerung der MP: Der Agent lernt durch Interaktion mit der MP als Umgebung, wobei er das Gelernte in einem NN speichert, welches er nach der Lernphase zur Steuerung nutzt.

Abschnitt 2.4.2 durch die notwendigen Prozesse in der MP. Damit löst der Agent von Anfrage zu Anfrage das Auftragsrouting direkt und das Stationsmanagement implizit, indem er die Stationen für die Produktionsschritte auswählt. Das Flottenmanagement erfolgt durch die direkte Koordination der Fahrzeuge. Auf Störungen reagiert der Agent in seinen Entscheidungen mithilfe der State-Informationen, die ihm die Umgebung zur Verfügung stellt. Die Durchsatzoptimierung erfolgt durch eine Durchlaufzeitminimierung mithilfe der von der Umgebung an den Agenten kommunizierten Rewards r_t.

Zur Erläuterung des in Abbildung 5.5 visualisierten lernenden Steuerungskonzepts betrachtet Abschnitt 5.3.1 zuerst ausführlich das Agentendesign, welches das Lösen der Entscheidungskategorien ermöglicht. Abschnitt 5.3.2 geht anschließend darauf ein, wie der Agent die Entscheidungskategorien adressiert.

5.3.1 Agentendesign

Die drei wesentlichen Design-Fragen beim DRL betreffen das State-, Action- und Reward-Design, damit der Agent die MP-Steuerung erfolgreich lernen kann. Abschnitt 5.1.3 verweist auf die Arbeiten von Kuhnle et al. [276], [281]–[283] und Altenmüller et al. [280] als Basis für die erfolgreiche Implementierung eines zentralen DRL-Agenten. Darauf stützt sich dieser Abschnitt, wobei die tatsächliche Umsetzung anwendungsspezifisch ist. Kuhnle et al. und Altenmüller et al. adressieren Halbleiterfertigungen. Für die MP ist das Design damit spezifisch anzupassen.

Die folgenden Absätze erläutern zuerst das gewählte Action-Design in der MP. Es folgt das State-Design, welches dem Agenten alle zum Lernen notwendigen Informationen zur Verfügung stellt. Das anschließende Reward-Design stellt die Durchsatzoptimierung sicher. Abschließend wird dargelegt, welches Setup der PPO-Agent nutzt, um Trajektorien zum Lernen zu sammeln.

Action-Design

Das Action-Design ergibt sich aus der Problemstellung in Abschnitt 2.4.2. Der PPO-Agent steuert die Fahrzeuge (Flottenmanagement), wodurch er die gesamte MP koordiniert. Wenn ein FTF seinen aktuellen Fahrauftrag abgearbeitet hat, fragt es beim Agenten nach einer neuen Action a_t. Damit ergeben sich Anfragen nach Actions an drei unterschiedlichen Orten in der MP:

- an der Quelle, entweder im Initialzustand der Produktion zum Starten eines Produkts (vgl. Abschnitt 5.3.2) oder für die erste anzufahrende Station eines gestarteten Produktionsauftrags,
- an einer Station M_k für den darauffolgenden Produktionsschritt des geladenen Auftrags, wobei ein Fahrzeug an der Station auf sein zugewiesenes Produkt wartet, bis er eine Anfrage startet und
- an der Senke nach Abschluss eines Produktionsauftrags für eine Entscheidung, welcher neue Auftrag an der Quelle abgeholt werden soll.

Abbildung 5.6 stellt dies schematisch durch rote Markierungen der Action-Anfragen dar.

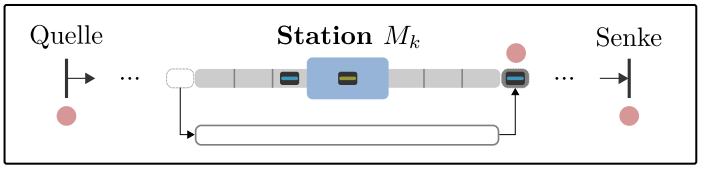


Abbildung 5.6: Rot markierte Action-Anfragen an den PPO-Agenten in der MP.

Die Actions, die der PPO-Agent in der in Abschnitt 2.4.2 beschriebenen MP damit durchführen kann, sind in der Menge $\mathcal{A}$ in (5.37) beschrieben, wobei der Index die jeweilige Destination des Fahrzeugs angibt:

$$\mathcal{A} := \left\{ A_{\text{Quelle}_x}, ..., A_{M_k}, ..., A_{\text{Senke}} \right\}. \tag{5.37}$$

Abschnitt 2.4.2 führt ein, dass ein FTF ein Produkt durch die gesamte MP begleitet. Hat eine derartige Zuweisung bei der Auftragseinlastung bereits stattgefunden, führt der

Agent das Produkt Schritt für Schritt zu Stationen M_k, beschrieben durch die Actions A_{M_k} (Stationsmanagement und Auftragsrouting). Sind alle Prozesse abgearbeitet, transportiert das Fahrzeug das geladene Produkt zur Senke der MP, was durch die Action A_{Senke} abgebildet ist. Hat ein FTF kein Produkt zugewiesen, kann der Agent ein neues Produkt PT_x an der Quelle starten, abgebildet durch Action A_{Quelle_x}. Damit existiert für jedes zu fertigende Produkt eine Action an der Quelle, wodurch die Auftragseinlastung gesteuert wird. Für eine MP mit 25 Stationen und zwei Produkttypen ist $\mathcal{A}$ damit definiert als:

$$\mathcal{A} := \left\{ A_{\text{Quelle}_1}, A_{\text{Quelle}_2}, A_{M_1}, ..., A_{M_{25}}, A_{\text{Senke}} \right\}. \tag{5.38}$$

Das obige Beispiel zeigt 28 Actions, die dem Agenten bei jeder Entscheidung zur Verfügung stehen. Bei Anfrage eines Fahrzeugs ist für den aktuellen State s_t allerdings lediglich eine Teilmenge $A_{t,\text{val}} \subseteq \mathcal{A}$ von zum Beispiel zwei dieser 28 Actions realistisch, da beispielsweise zwei Stationen für den nächsten Produktionsschritt des geladenen Produkts zur Verfügung stehen. Die anderen 26 Actions führen für diesen Zeitschritt zu nicht erwünschten Produktionssituationen. Daraus ergeben sich zwei Nachteile: Erstens müsste der Agent über seine Erfahrung lernen, welche Actions in jedem Zeitschritt t valide und welche es nicht sind. Zweitens ist das zu lernende Problem dadurch komplexer. May et al. [290] wenden hierfür Action Masking an.

Action Masking verhindert die Selektion invalider Actions durch Manipulation des Policy-Netzes. Hierbei werden die Action-Wahrscheinlichkeiten $\pi_{\boldsymbol{\theta}}(a_t \mid s_t)$ der in s_t invaliden Actions a_t zu nahezu null manipuliert. Dadurch wählt sie der Agent nicht aus. Er kann sich auf das Wesentliche konzentrieren, was die Performanz und Lerngeschwindigkeit erhöht [302], [303]. Huang und Ontanon [302] heben hierbei auch die Skalierbarkeit durch Action Masking hervor. Im Vergleich zu Konfigurationen, in denen der Agent über stark negative Bestrafungs-Rewards lernt, valide und invalide Actions zu unterscheiden, lernt der Agent mit Action Masking schneller bessere Policies. Einen derartigen Vergleich zu den Vorteilen des Action Maskings führt auch Abschnitt 6.3.3 durch.

Die Actions a_t wählt der Agent auf Basis des aktuellen States s_t, welcher auch Informationen zu validen und nicht validen Actions beinhaltet.

State-Design

Wie in Abschnitt 5.2.1 eingeführt, muss der State alle Informationen aus der Vergangenheit der Agent-Umgebungs-Interaktion beinhalten, die für die Zukunftsentscheidungen relevant sind, um die Markov-Eigenschaft zu erfüllen. Der State s_t in Zeitschritt t informiert den Agenten hierfür über den aktuellen Zustand der Quelle, den Zustand der einzelnen Stationen, den Zustand des anfragenden Fahrzeugs, den Fortschritt des geladenen Produkts sowie die validen Actions für die aktuelle Anfrage. In Abhängigkeit der darzustellenden Information verwendet die Arbeit hierbei wie auch Kuhnle et al. [276], [281]–[283] und Altenmüller

et al. [280] „One-Hot-Encoding“, wobei die Informationen binär codiert werden. One-Hot-Encoding hat sich außerdem in den Arbeiten von Waschneck et al. [277], [287], [288] als effektiv erwiesen.

Der Zustand der Quelle betrifft das umzusetzende Produktionsprogramm der MP. Nach Abschnitt 2.4.2 ist hierbei ein vorgegebenes periodisches Produktionsprogramm abzuarbeiten, welches den Produktmix abbildet, beispielsweise: $A \rightarrow B \rightarrow A \rightarrow \ldots$ etc. Der Agent muss deshalb sicherstellen, dass dieses Verhältnis (hier: 1:1) umgesetzt wird. Dafür beinhaltet der State s_t für jeden zu fertigenden Produkttypen PT_x eine Information sq_x. Der Wert der Variable sq_x ist entweder 0 oder 1 (vgl. (5.39)). Bei Initialisierung werden alle Werte auf 1 gesetzt. Der Agent kann sich dann entscheiden, welches Produkt er startet, wonach der Wert des gestarteten Produkts 0 beträgt. Erst wenn er jeden Produkttypen einmal gestartet hat, setzt die Umgebung alle Werte wieder auf 1. So bleibt der gewollte Produktmix erhalten, wobei die Einlastungssequenz der Produkttypen in Abhängigkeit der Agentenentscheidungen pro Einlastungskontingent variiert.

$$sq_x := \begin{cases} 1 & \text{falls Produkt in Periode noch nicht gestartet,} \\ 0 & \text{sonst.} \end{cases} \tag{5.39}$$

Zur Darstellung der Stationen beinhaltet s_t pro Station M_k zwei Informationen: die Stationsbelegung $m_{k,\mathrm{WZ}}$ an der Station sowie Informationen über eine potenzielle Störung $m_{k,\mathrm{stoer}}$.

Die Stationsbelegung $m_{k,\mathrm{WZ}}$ beziffert die Wartezeit, die an M_k mindestens vergeht, bis ein neuer, aktuell noch nicht eingeplanter Auftrag in Bearbeitung gehen kann. Zwei Fälle sind hier zu unterscheiden:

- Stationen sind **nicht valide Zielstationen**: Gesamtbild zu Stationsbelegungen aller Stationen, die nicht einer potenziell validen Zielstation entsprechen.

- Stationen sind **valide Zielstationen**: Information über die zur Wahl stehenden Zielstationen, wobei zusätzlich die Transportzeiten zu den Zielstationen einzuplanen sind.

Für alle zum Zeitpunkt t **nicht validen Zielstationen** bildet $m_{k,\mathrm{WZ}}$ das Maximum aus der Summe der dort aktuell eingeplanten Prozesszeiten $t_{ipl,k,\mathrm{Rest}} + \sum_{i=0}^{n_{J,M_k}} t_{ipl,k}$ und der Mindest-Transportzeit $t_{\mathrm{EP},min}$ im EP der Station. $t_{ipl,k,\mathrm{Rest}}$ bezeichnet hierbei die Rest-Bearbeitungszeit eines Auftrags, der sich an M_k in Bearbeitung befindet. Die Anzahl n_{J,M_k} umfasst die im EP oder vor der Station M_k wartenden bzw. sich auf dem Weg zur Station befindlichen Aufträge, deren Prozesszeiten $t_{ipl,k}$ an M_k ebenfalls ausständig sind. $t_{\mathrm{EP},min}$ muss berücksichtigt werden, weil bei Ankunft an einer Station immer mindestens noch die fließbandbedingte Puffer-Transportzeit vergeht, welche hier als Wartezeit interpretiert wird. Der zweite Fall betrifft nur die potenziellen Zielstationen des anfragenden Fahrzeugs. Handelt es sich um eine **valide Zielstation**, bildet $m_{k,\mathrm{WZ}}$ das Maximum aus der oben beschriebenen

Prozesszeitensumme und aus der Transportzeit einschließlich der Puffer-Transportzeit an der Zielstation ab. Die Erweiterung um die Transportzeit $t_{\mathrm{TR},ipl}$ zur Zielstation stellt sicher, dass die Transporte zur Entscheidung zwischen validen Actions Berücksichtigung finden. Ist die Wartezeit durch die Stationsbelegung geringer als die Transportzeit über FTF und EP zur Station, plant der Agent realistisch mit letzterem Term als dem frühestmöglichen Zeitpunkt, um dort in Bearbeitung zu gehen. Im Gegensatz zu dieser detaillierteren Sichtweise trägt der erste Fall ausschließlich zum Gesamtbild des Agenten bei, welches er über die aktuellen Stationsbelegungen der MP hat. Diese Informationen könnten ihm beispielsweise dienen, wenn er schon zukünftige Actions bei seiner aktuellen Wahl antizipiert bzw. einbezieht. Führt das anfragende Fahrzeug keinen Auftrag, setzt nur der erste Fall ein, da die Transportzeiten zu den Stationen aktuell irrelevant sind. Der Agent hat dann die Aufgabe, ein neues Produkt in der Quelle zu starten. Trotzdem ermöglicht das Gesamtbild zu den Stationsbelegungen auch hier eine Antizipation möglicher zukünftiger Actions. Die Normalisierung der State-Variablen mit t_{avg} erklärt ein späterer Absatz beim Thema des Reward-Designs.

$$m_{k,\mathrm{WZ}} := \begin{cases} \frac{\max(t_{\mathrm{EP},min},\, t_{ipl,k,\mathrm{Rest}} + \sum_{i=0}^{n_{J,M_k}} t_{ipl,k})}{t_{\mathrm{avg}}} & \text{falls } M_k \text{ kein valides Ziel,} \\ \frac{\max(t_{\mathrm{TR},ipl} + t_{\mathrm{EP},min},\, t_{ipl,k,\mathrm{Rest}} + \sum_{i=0}^{n_{J,M_k}} t_{ipl,k})}{t_{\mathrm{avg}}} & \text{sonst.} \end{cases} \tag{5.40}$$

Die zweite Information zum Zustand einer Station $m_{k,\mathrm{stoer}}$ stellt dar, ob sie sich in Störung befindet und falls sie gestört ist, wie lange diese Störung schon andauert. Der Wert $t_{\mathrm{jetzt}} - t_{\mathrm{stoer,start}}$ gibt die Zeit der Störung an, indem die Differenz aus der aktuellen Zeit t_{jetzt} und dem Störungsstart $t_{\mathrm{stoer,start}}$ an der Station berechnet wird. Die 1 wird zusätzlich berücksichtigt, um eine Störung klar von einer nicht vorhandenen Störung zu unterscheiden. Liegt $\frac{t_{\mathrm{jetzt}} - t_{\mathrm{stoer,start}}}{t_{\mathrm{avg}}}$ aufgrund einer gerade begonnenen Störung noch nahe der 0, signalisiert die 1 trotzdem die Störung. Je länger die Störung dauert, desto größer wird $m_{k,\mathrm{stoer}}$. Diese Information zur aktuellen Störsituation ermöglicht dem Agenten einen effektiven Umgang mit Störungen (Störungsmanagement). Wiederum erfolgt eine Normalisierung mit t_{avg}.

$$m_{k,\mathrm{stoer}} := \begin{cases} 1 + \frac{t_{\mathrm{jetzt}} - t_{\mathrm{stoer,start}}}{t_{\mathrm{avg}}} & \text{falls Störung an Station } M_k, \\ 0 & \text{sonst.} \end{cases} \tag{5.41}$$

Der Zustand des anfragenden Fahrzeugs wird von zwei Informationen repräsentiert. vpt_x gibt an, ob das Fahrzeug Produkttyp PT_x geladen hat. Wenn kein zugewiesenes Produkt existiert, sind alle Positionen vpt_x auf 0 gesetzt. vp_y gibt an, wo sich das anfragende Fahrzeug befindet. Hier kommen die Quelle, die Senke sowie alle Stationen infrage.

$$vpt_x := \begin{cases} 1 & \text{falls } PT_x \text{ auf dem Fahrzeug,} \\ 0 & \text{sonst.} \end{cases} \tag{5.42}$$

$$vp_y := \begin{cases} 1 & \text{falls Fahrzeug an Ort } y, \\ 0 & \text{sonst.} \end{cases} \tag{5.43}$$

Der Fortschritt des geladenen Produkts zeigt sich in pf_l. Hierbei muss der Agent die maximale Anzahl an Prozessen aller zu fertigenden Produkte als Neuronen abbilden, damit alle Produkte berücksichtigt werden können. Hat das anfragende Fahrzeug einen Produktionsauftrag geladen, wird der aktuell anstehende Produktionsschritt O_{ipl} des geladenen Produkts J_i auf $pf_l = 1$ gesetzt, alle anderen stehen auf 0. Ist das Fahrzeug auftragsfrei, sind alle potenziellen Schritte zu 0 gesetzt.

$$pf_l := \begin{cases} 1 & \text{falls } l \text{ die Nummer der aktuell anstehenden Operation } O_{ipl} \text{ von } J_i \text{ ist,} \\ 0 & \text{sonst.} \end{cases} \tag{5.44}$$

Letztlich enthält der State die Informationen va_z zur Validität der verfügbaren Actions $\mathcal{A}$. $A_{t,\text{val}} \subseteq \mathcal{A}$ gibt hierbei die Teilmenge der validen Actions im aktuellen Zeitschritt t an. Eine valide Action erhält eine 1, eine nicht valide Action eine 0. Das implementierte Action Masking nutzt diese Informationen. Außerdem kann der Agent mit diesen Informationen schlussfolgern, ob er eine invalide Action gewählt hat, falls Action Masking inaktiv ist.

$$va_z := \begin{cases} 1 & \text{falls } A_z \in A_{t,\text{val}}, \\ 0 & \text{sonst.} \end{cases} \tag{5.45}$$

Abschließend lässt sich ein State s_t für eine MP mit 25 Stationen und zwei Produkttypen mit jeweils 14 Bearbeitungsschritten beispielhaft zusammenfassen als:

$$s_t := [\overset{\text{Quelle}}{\underset{sq_1}{1}\ \underset{sq_2}{0}}\ \overset{M_1}{\underset{m_{1,\text{WZ}}}{0.8}\ \underset{m_{1,\text{stoer}}}{0}}\ \cdots\ \overset{M_{25}}{\underset{m_{25,\text{WZ}}}{0.3}\ \underset{m_{25,\text{stoer}}}{1.3}}\ \overset{V_j\ \text{Produkt}}{\underset{vpt_1}{0}\ \underset{vpt_2}{1}}\ \overset{V_j\ \text{Position}}{\underset{vp_1}{1} \cdots \underset{vp_{27}}{0}}\ \overset{J_i\ \text{Fortschritt}}{\underset{pf_1}{1} \cdots \underset{pf_{14}}{0}}\ \overset{\text{Valide Actions}}{\underset{va_1}{0} \cdots \underset{va_{28}}{0}}\ ].$$

Der Wert $sq_1 = 1$ zeigt an, dass Produkttyp PT_1 in der Quelle zur Abholung bereitliegt. Der Wert $sq_2 = 0$ legt fest, dass Produkttyp PT_2 aktuell nicht gestartet werden kann. Station M_1 hat eine normalisierte Wartezeit $m_{1,\text{WZ}}$ von 0.8 und befindet sich nicht in Störung: $m_{1,\text{stoer}} = 0$. M_{25} unterliegt hingegen einer Störung: $m_{25,\text{stoer}} = 1.3$. Das anfragende Fahrzeug V_j trägt den Produkttypen PT_2 ($vpt_2 = 1$) und befindet sich auf Position $vp_1 = 1$ und damit an der Quelle. Das bedeutet, das geladene Produkt wurde soeben gestartet ($sq_2 = 0$) und es steht die Entscheidung für den ersten Produktionsschritt an, was auch $pf_1 = 1$ zeigt. Über die validen Actions kann sich der Agent nun für den ersten Schritt entscheiden, indem er das Fahrzeug, auf dem das Produkt geladen ist, zu einer validen Zielstation schickt.[5]

Damit der Agent auf Basis der gewählten Actions a_t in Input-States s_t lernen kann, benö-

[5]Die 27 möglichen Fahrzeugpositionen ergeben sich aus 25 Stationen sowie der Quelle und der Senke der MP. Daraus resultieren 28 mögliche Actions, weil pro Produkttyp eine Action für die Quelle vorgesehen ist.

tigt er Rewards r_t, die ihm anzeigen, welche Actions zum gewünschten Ergebnis führen und welche nicht.

Reward-Design

Nach Abschnitt 2.4.2 strebt die MP eine Durchsatzmaximierung an. Bär et al. [289] maximieren den Durchsatz, indem ein Agent für die Produkte kürzere Warteschlangen an Stationen bevorzugt, wenn er sich für den nächsten Produktionsschritt entscheidet. Dadurch werden die Durchlaufzeiten der Aufträge minimiert, was den Durchsatz maximiert. Dieses Konzept setzt auch diese Arbeit um. Da ein FTF ein Produkt durch die gesamte Produktion trägt und der Agent direkt die Fahrzeuge steuert, hat er durch seine Entscheidungen direkten Einfluss auf die Auftragsdurchlaufzeiten. Durch ein Reward-Signal, welches in der Produktion vergangene Zeiten negativ übergibt, wird der Agent angeregt, Zeitverschwendung im System und für die Aufträge zu vermeiden. Der Reward wird als negativ übergeben, weil der PPO-Agent nach Abschnitt 5.2.3 das NN in einer Maximierungsaufgabe per Gradientenaufstieg optimiert. Das Reward-Design ergibt sich zu:

$$r_t := \begin{cases} r_{\text{punish}} & \text{falls } a_t \notin A_{t,\text{val}}, \\ \frac{-t_{\text{TR},Pos\rightarrow Senke}}{t_{\text{avg}}} + r_{\text{bonus}} & \text{falls } a_t == A_{\text{Senke}} \in A_{t,\text{val}}, \\ \frac{-t_{\text{TR},Pos\rightarrow Quelle}}{t_{\text{avg}}} & \text{falls } a_t == A_{\text{Quelle},x} \in A_{t,\text{val}}, \\ \frac{-(t_{\text{TR},ipl}+max(t_{\text{WZ},ipl},\, t_{\text{EP},min}))}{t_{\text{avg}}} & \text{sonst,} \end{cases} \tag{5.46}$$

wobei der **erste Fall** nur eintreten kann, falls Action Masking nicht aktiv ist. In diesem Fall wird eine invalide Action mit einem hochnegativen Reward r_{punish} bestraft und nicht ausgeführt. Der Agent muss dann auf Basis des gleichen States eine neue Entscheidung treffen. Das Wissen über valide und nicht valide Actions auf Basis des States wird dem Agenten implementierungsseitig zur Verfügung gestellt. Der **zweite Fall** betrifft die Fertigstellung eines Auftrags, wenn er von einem FTF zur Senke gebracht wird. Hier bekommt der Agent die negative Transportzeit $-t_{\text{TR},Pos\rightarrow Senke}$ von seiner aktuellen Position aus als Reward. Zusätzlich wird ein positiver Bonus-Reward r_{bonus} gegeben, der dem Agenten signalisiert, dass die Fertigstellung eines Auftrags im Sinne einer Durchsatzmaximierung positiv zu bewerten ist. Der **dritte Fall** betrifft die Leerfahrt eines Fahrzeugs zur Quelle, um ein neues Produkt aufzunehmen, wobei der Agent wiederum die negative Transportzeit $-t_{\text{TR},Pos\rightarrow Quelle}$ als Reward sieht. Der **vierte Fall** stellt den Standardfall dar. Der Agent führt eine valide Action $a_t \in A_{t,\text{val}}$ für ein aufgeladenes Produkt aus. Hierbei berechnet sich r_t aus der vergangenen Transportzeit $t_{\text{TR},ipl}$ für die anvisierte Station sowie der vergangenen Zeit im EP, welche entweder der Mindest-Transportzeit $t_{\text{EP},min}$ oder der Wartezeit $t_{\text{WZ},ipl}$ im Puffer entspricht. Letztere wird relevant, wenn der Auftrag über seine Puffer-Transportzeit hinaus aufgrund der Stationsbelegung warten muss. Wiederum wird die Zeit negativ übergeben.

Der vierte Fall führt zur auftragsbasierten Durchlaufzeitminimierung, weil der Agent dadurch beginnt, Transportzeiten und Wartezeiten zu minimieren. Unter Berücksichtigung der State-Informationen $m_{k,\mathrm{WZ}}$ zu den aktuellen Belegungen aller Stationen M_k kann der Agent auch schlussfolgern, woher Wartezeiten und die damit verbundenen Durchlaufzeitverlängerungen kommen. Über die Erfahrungen mithilfe der Interaktion, wobei der Agent Actions aus States mit den Rewards verknüpft, lernt er geringere Durchlaufzeiten zu erzeugen und maximiert damit den Durchsatz. Wie beim State werden auch beim Reward alle Zeiten mit t_{avg} normalisiert. Das dargestellte Reward-Design hat sich in Anlehnung an [289] als effektiv erwiesen, um eine Durchsatzmaximierung zu erreichen, wie Kapitel 6 zeigen wird.

Die Normalisierung mit t_{avg} ist durchzuführen, damit die Zeiten in den State-Informationen $m_{k,\mathrm{WZ}}$ und $m_{k,\mathrm{stoer}}$ sowie im Reward r_t nicht 1:1 in die Berechnungen des Agenten eingehen. Im Vergleich zu den one-hot-codierten Informationen nehmen sie sonst verhältnismäßig große Werte an. t_{avg} skaliert die Zeit in Anlehnung an [283] hin zu kleineren Werten und berechnet sich zu:

$$t_{\mathrm{avg}} = \frac{n_V - 1}{n_M} \cdot \frac{1}{n_J} \cdot \sum_{i=1}^{n_J} (\frac{1}{n_O} \cdot \sum_{l=1}^{n_O} t_{i1l}) + \frac{1}{{n_M}^2} \cdot \sum_{k1=1}^{n_M} \sum_{k2=1}^{n_M} t_{\mathrm{TR},M_{k1} \rightarrow M_{k2}}, \tag{5.47}$$

wobei der gesamte Term eine Annäherung an die mittlere erwartete Übergangszeit zwischen zwei Stationen berechnet, die ein Auftrag zwischen zwei Operationen verbringt. Dadurch resultieren Werte für $m_{k,\mathrm{WZ}}$, $m_{k,\mathrm{stoer}}$ sowie r_t, die sich näher an der null und damit in derselben Größenordnung wie die one-hot-codierten Informationen befinden. Der erste Term berechnet die mittlere Bearbeitungszeit aller Operationen über alle Aufträge: $\frac{1}{n_J} \cdot \sum_{i=1}^{n_J} (\frac{1}{n_O} \cdot \sum_{l=1}^{n_O} t_{i1l})$, wobei die Zeiten als potenzielle Wartezeiten durch $\frac{n_V - 1}{n_M}$ an die Stationen verteilt werden. Damit die durchschnittliche Produktionssituation abgebildet wird, gilt hierfür die Annahme, dass sich die n_V Fahrzeuge mit ihren Produkten gleichmäßig auf die n_M Stationen verteilen. Von n_V wird hierbei ein Fahrzeug subtrahiert, weil in jeder Produktionssituation ein Fahrzeug die aktuelle Action-Anfrage stellt. Befinden sich beispielsweise $n_V = 20$ Fahrzeuge in einer MP mit $n_M = 25$ Stationen, ergibt sich unter der Annahme einer durchschnittlichen Bearbeitungszeit von $\frac{1}{n_J} \cdot \sum_{i=1}^{n_J} (\frac{1}{n_O} \cdot \sum_{l=1}^{n_O} t_{i1l}) = 50s$ die durchschnittliche Wartezeit zu $\frac{20-1}{25} \cdot 50s = 38s$. Der zweite Term berechnet die durchschnittliche Transportzeit zwischen allen n_M Stationen, wobei $t_{\mathrm{TR},M_{k1} \rightarrow M_{k2}}$ die Transportzeit zwischen M_{k1} und M_{k2} bezeichnet. Die Mindest-Transportzeiten in den Puffern werden hierbei vernachlässigt. Insgesamt ergibt sich aus der durchschnittlichen Wartezeit (Term 1) und der durchschnittlichen Transportzeit (Term 2) eine geschätzte Übergangszeit, die für die gewünschte Skalierung sorgt. t_{avg} ist hierbei einmalig zu berechnen und bleibt für die gesamte Trainingsphase konstant.

Der folgende Abschnitt erläutert, wie der PPO-Agent das Action-, State- und Reward-Design zum Lernen nutzt und in die Tat umsetzt.

Setup zum Lernen und Sammeln der Trajektorien

Abbildung 5.7 stellt überblicksweise das Setup des in der vorliegenden Arbeit entwickelten PPO-Agenten dar. Es handelt sich um das geteilte NN des Actors und Critics (vgl. Abschnitt 5.2.3). Als Input des NNs dient der aktuelle State s_t der Umgebung. Der generierte Output des Actors gibt Action-Wahrscheinlichkeiten $\pi(a_t \mid s_t)$ nach (5.8) aus, welche anzeigen, wie sicher sich der Agent ist, dass eine bestimmte Action $a_t \in \mathcal{A}$ im aktuellen State s_t die optimale Wahl darstellt. Action Masking manipuliert hierbei die Wahrscheinlichkeiten $\pi(a_t \mid s_t)$ nicht valider Actions zum Zeitpunkt t in Richtung null. Der generierte Output des Critics beläuft sich auf die Value-Funktion $v_\pi(s_t)$ nach (5.9) des aktuellen States s_t. Die Größen der Input- und Output-Schicht h_{in} und h_{out} sind dabei problemspezifisch und hängen vom State- und Action-Design ab. Die Anzahl der versteckten Schichten sowie die Anzahl der Neuronen pro versteckter Schicht sind entsprechend zu definieren.

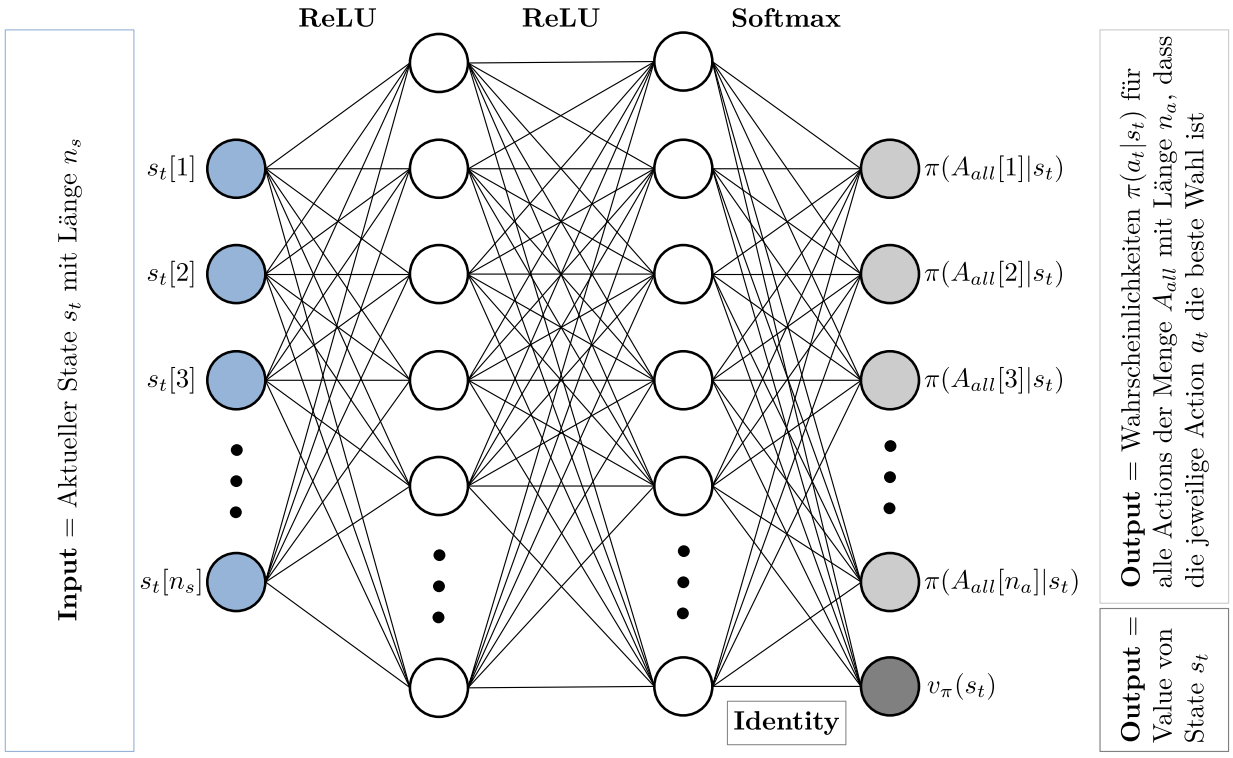


Abbildung 5.7: Darstellung des NNs des PPO-Agenten, welches den aktuellen State s_t als Input bekommt und Action-Wahrscheinlichkeiten $\pi(a_t \mid s_t)$ sowie die Value-Funktion $v_\pi(s_t)$ als Output berechnet.

Nach Alg. 5.1 durchläuft AC-PPO in der Lernphase n_{episodes} Episoden. In jeder Episode sammelt der Agent n_τ State-Action-Reward-Tripel nach dem Interaktionszyklus aus Abbildung 5.1. Für jede Action-Anfrage aus der Umgebung wählt der Agent hierbei eine zufällige Action a_t nach der vom NN berechneten Wahrscheinlichkeitsverteilung für State s_t in Zeitschritt t. Damit die Output-Schicht für die Actions eine Wahrscheinlichkeitsverteilung ergibt, wird die „Softmax“-Aktivierungsfunktion angewendet. Den Critic-Output $v_\pi(s_t)$ benötigt der Agent später für das NN-Update. Im Gegensatz zu den Action-Wahrscheinlichkeiten mit

Softmax, findet für den Value die „Identity“-Aktivierungsfunktion Anwendung. Alle anderen Schichten nutzen „Leaky Rectified Linear Units“ (Leaky ReLU) als Aktivierungsfunktion. Weiterführende Informationen zu Aktivierungsfunktionen finden sich beispielsweise in [291, S. 11ff.]. Die nach der Wahrscheinlichkeitsverteilung gewählte Action a_t wird von der Umgebung ausgeführt. Mit der nächsten Action-Anfrage eines Fahrzeugs übergibt die Umgebung einen Reward r_{t+1} sowie den neuen State s_{t+1}.

Die von der Umgebung zurückgegebenen Rewards nach (5.46) entsprechen hierbei FTF-spezifischen Rewards, damit sie die Durchlaufzeiten der Aufträge repräsentieren und minimieren. Das heißt, in der Umgebung verwaltet jedes Fahrzeug seine individuellen Rewards und berechnet die Rewards relativ zur vorherigen fahrzeugspezifischen Action-Anfrage. Bei jeder Action-Anfrage übergibt die Umgebung dann den fahrzeugspezifischen Reward des anfragenden Fahrzeugs. Dabei ist wichtig, dass dieser Reward in der Trajektorie der richtigen Action für das spezifische Fahrzeug aus einem definierten State zugeordnet wird, auch bekannt als „Credit Assignment Problem“ [179, S. 11]. Dies geschieht nicht automatisch, weil die Fahrzeuge in der Umgebung ereignisorientiert Anfragen stellen, die in beliebiger Reihenfolge erfolgen können. Da der Agent n_V Fahrzeuge gleichzeitig koordiniert, ist eine Action a_t für ein spezifisches Fahrzeug meist noch nicht abgeschlossen, wenn ein anderes FTF eine Anfrage stellt. Das heißt, der zurückgegebene Reward r_{t+1} ist aus der zentralen Agentensicht nicht immer der vorangegangenen Action a_t aus State s_t zuzuordnen, weil a_t beispielsweise für das Fahrzeug V_1 ausgeführt wurde, r_{t+1} hingegen dem Reward für die letzte Action von Fahrzeug V_5 entspricht. Demnach sind die Rewards nachträglich zu sortieren, bildlich dargestellt in Abbildung 5.8. Ein X repräsentiert den Platzhalter für einen Reward. Der Action-Anfrage von V_1 in s_t folgt eine Anfrage von V_5 in s_{t+1}. Dabei übergibt die Umgebung einen Reward r_{t+1}, welcher nicht a_t von V_1, sondern der vorangegangenen Action für V_5 zuzuordnen ist (nicht sichtbar in Abbildung). Der angedeutete Pfeil stellt schematisch dar, wie diese nachträgliche Zuordnung passiert, indem die vorangegangene Action von V_5 gesucht und der Reward r_{t+1} dort in die Trajektorie eingeordnet wird. Gleiches gilt beispielsweise für

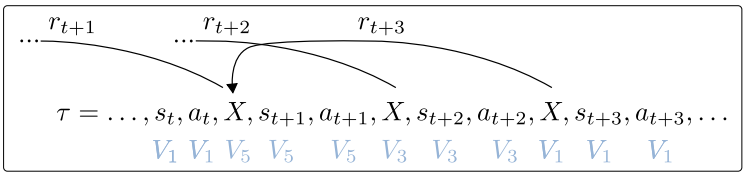


Abbildung 5.8: FTF-spezifische Einordnung der von der Umgebung zurückgegebenen Rewards.

den Reward r_{t+3} von V_1, welchen die Umgebung für das Fahrzeug in State s_{t+3} übergibt. r_{t+3} ist wiederum der vorangegangenen Action a_t von Fahrzeug V_1 zuzuordnen, weshalb das X nach Action a_t mit dem Reward-Wert von r_{t+3} auszutauschen ist[6]. Durch diese Vorgehens-

[6]Die Zeitschritte verschwimmen dadurch ineinander. Der in $t+3$ zurückgegebene Reward ist t zuzuordnen. Die Zeitschritt-Bezeichnungen verlieren damit für die Rewards ihre Aussagekraft. Wichtig ist die korrekte Einordnung des Reward-Werts bei der Action a_t, die zum Reward führt.

weise verknüpft der Agent die fahrzeugspezifischen Rewards mit den fahrzeugspezifischen Actions. Er führt das „Credit Assignment“ durch.

Auf dieser Grundlage ist die Interaktion mit der Umgebung pro Episode so lange zu wiederholen, bis sich n_τ korrekt zugeordnete State-Action-Reward-Tripel gebildet haben. Diese Bedingung muss für alle n_{env} parallelen Trajektorien, die der Agent in den Umgebungen sammelt, gegeben sein. Dadurch stehen dem Agenten für das NN-Update nach jeder Episode korrekt zugeordnete Daten zur Verfügung. Die State-Action-Reward-Tripel, deren korrekte Zuordnung noch aussteht, bleiben als Erfahrungen pro Umgebung gespeichert und gehen in die nächsten Episoden ein.

Zusätzlich zum Sammeln vollständig sortierter Trajektorien der Länge n_τ sind richtig sortierte State-Action-Reward-Tripel aus der Zukunft vorzuhalten, weil die Advantages in (5.24) einen Blick auf die Folgerewards der aktuell bewerteten Action a_t werfen. Fände kein Vorhalten zukünftiger Tripel statt, wäre die Advantage-Berechnung der letzten Actions einer Trajektorie fehlerhaft. Der PPO-Agent sammelt also nicht nur n_τ korrekte State-Action-Reward-Tripel. Er sammelt pro Episode noch weitere n_{zuk} zukünftige korrekte Tripel zur Berechnung der Advantages. Diese Zukunfts-Tripel gehen dann auch nach jedem Update als bereits gesammelte Erfahrungen in die nächste Trajektorie ein. Der Agent ist hierbei so konzipiert, dass die n_{zuk} Zukunfts-Tripel so weit reichen, bis der prozentuale Einfluss über die Discount-Faktoren bei $< 1 \cdot 10^{-6}$ liegt. Dieser Wert wurde empirisch ermittelt.

Sind obige Bedingungen erfüllt, nutzt der Agent die korrekten n_τ Erfahrungen[7] der parallelen Trajektorien nach Alg. 5.1, um das NN mittels Gradientenaufstieg zu aktualisieren. Zum Parameter-Update nutzt der Agent hierfür den im DRL häufig angewendeten (beispielsweise in [274], [295], [296]) Optimierer „Adam“ [304], ebenfalls empfohlen von Ruder [305].

Nach dem Update folgt der Agent der neuen Policy $\pi_{\boldsymbol{\theta}'}$ und sammelt erneut $n_\tau + n_{\text{zuk}}$ korrekt zugeordnete State-Action-Reward-Tripel in n_{env} parallelen Umgebungen, um das NN zu aktualisieren. Jede Trajektorie startet dabei im letzten State der vorhergehenden Trajektorie. Es handelt sich daher um eine Continuing-Task, eingeführt in Abschnitt 5.2.1. Dies spiegelt eine Produktion wider, die 24 Stunden am Tag, sieben Tage die Woche produziert und damit nicht regelmäßig von Neuem zu initialisieren ist.

Nach Abschluss der obigen Lernphase verwendet der PPO-Agent seine Erfahrung zum Steuern der MP in den fünf Entscheidungskategorien.

5.3.2 Lösen der fünf Entscheidungskategorien

Abschnitt 5.3.1 deutet bereits an, wie der PPO-Agent die fünf Entscheidungskategorien der MP lernt und löst. In der an die Lernphase anschließenden Steuerungsphase führt der Agent das Gelernte aus, indem er bei Action-Anfragen die optimale Action mit der höchsten Wahr-

[7] Die n_{zuk} Zukunfts-Tripel sind lediglich für die Advantage-Berechnungen notwendig. Sie gehen nicht in das NN-Update ein.

scheinlichkeit wählt. Der Agent steuert dabei direkt die Fahrzeuge. Diese fragen den Agenten hierbei ereignisbasiert an, wenn sie einen neuen Fahrauftrag benötigen, entweder zum Transport des ihnen zugewiesenen Produkts oder als Leerfahrt zum Starten eines neuen Produkts. Die folgenden Absätze fassen darauf basierend die Lösung der fünf Entscheidungskategorien zusammen.

Auftragseinlastung

Die Randbedingungen bei der Auftragseinlastung im lernenden Ansatz sind identisch zu denen der zentralen und dezentralen Steuerung in Abschnitt 3.3.1 und Abschnitt 4.3.1. Der Unterschied liegt in der Festlegung der Einlastungszeitpunkte sowie in der Umsetzung des Produktmixes. Hierbei wird beim lernenden Ansatz keine feste Einlastungsreihenfolge vorgegeben, sondern nur der Produktmix beibehalten, wie die folgenden Absätze zeigen. Der Vollständigkeit und Lesbarkeit halber werden die überschneidenden Aspekte mit dem zentralen und dezentralen Ansatz hier erneut beschrieben.

Der PPO-Agent geht nach den in Abschnitt 2.4.2 getroffenen Annahmen von einem vordefinierten Produktmix aus. Dieser legt die Anzahl der zu fertigenden Produkte pro Produkttyp fest und in welchem Verhältnis sie einzulasten sind. Zu treffen sind Entscheidungen über die Zuweisung von WTs und FTFs sowie über die Zeitpunkte der Einlastung der einzelnen Produktionsaufträge. Eine erfolgreiche Einlastung erfordert deshalb die Verfügbarkeit eines WTs und eines FTFs, auf denen das Produkt im System gestartet werden kann. Die Anzahl verfügbarer WTs bzw. FTFs ($n_{\mathrm{WT}} = n_V$, Annahme in Abschnitt 2.4.2) begrenzt den Produktionsbestand in der MP zu: $PB_{max} = n_{\mathrm{WT}} = n_V$. Hierfür bietet sich eine Orientierung an der in Abschnitt 2.2.1 vorgestellten einfachen Conwip-Steuerung an, welche einen neuen Auftrag einlastet, wenn gilt: $PB < PB_{max}$ [74, S. 373-385]. Mit diesem Hintergrund, ergänzt mit einigen problemspezifischen Erweiterungen, folgt die lernende Auftragseinlastung dem Produktionsbestand und basiert auf zwei Prinzipien:

1. **Initiale Einlastung**: Im Initialzustand der Produktion stehen n_{WT} WTs und n_V Fahrzeuge hintereinander in der Quelle der MP bereit. Sind beispielsweise drei Produkttypen im Verhältnis 1:1:1 zu produzieren, hat der Agent die Wahl zwischen allen Produkttypen durch die State-Informationen $sq_1 = 1$, $sq_2 = 1$ und $sq_3 = 1$. Bei Start eines Produktes reduziert sich sq_x zu 0. Wurde jeder Produkttyp im vorgegebenen Mix einmal gestartet, steht ein neues Kontingent im Verhältnis 1:1:1 zur Verfügung. Die tatsächliche Auftragseinlastung erfolgt ähnlich zu [283]. Ein Auftrag wird durch die Entscheidung des PPO-Agenten, ein FTF zur Quelle zu schicken, eingelastet. Mit der Action A_{Quelle_x} entscheidet der Agent dann auch, welcher der zur Verfügung stehenden Produkttypen einzulasten ist, abhängig von den gegenwärtigen State-Informationen sq_x. Die in der Quelle verfügbaren FTFs starten in der initialisierten Reihenfolge ihre

Action-Anfragen. Die erste valide Action jedes Fahrzeugs entspricht einer Einlastungs-Action A_{Quelle_x}. Bei der Einlastung wird dem Fahrzeug auch ein verfügbarer WT in der in der Quelle gegebenen Reihenfolge zugewiesen. Damit werden also die verfügbaren n_{WT} WT-FTF-Paare (WT_1/V_1, WT_2/V_2 etc.) den ersten Produktionsaufträgen des Produktionsprogramms entsprechend der Actions A_{Quelle_x} zugeordnet. Der tatsächliche Zeitpunkt der Auftragseinlastung S_{J_i} des Auftrags J_i entspricht dem Zeitpunkt, zu dem das FTF am Einlastungsort in der Quelle ankommt. Da sich die Fahrzeuge und WTs in der Initialisierungsphase bereits in der Quelle befinden, reduziert sich diese Zeit implementierungsbedingt zu $1s$.

2. **Bestandsgeregelte Einlastung**: Sind alle WTs und FTFs in der MP im Einsatz, gilt: $PB = PB_{max}$. Der PPO-Agent wartet dann auf ein verfügbar werdendes WT-FTF-Paar durch die Fertigstellung eines Produktes, wodurch PB unter die Bestandsgrenze PB_{max} fällt. Das Fahrzeug, welches an der Fertigstellung des Produktes in der Senke beteiligt ist, initiiert eine neue Action-Anfrage beim Agenten, wobei wieder nur eine der Einlastungs-Actions A_{Quelle_x} valide ist. Unter Beibehaltung des Produktmixes erfolgt dann die Bereitstellung eines neuen Produktionsauftrags, sobald der WT und das FTF in der Quelle der MP ankommen. Wiederum entspricht der letztendliche Einlastungszeitpunkt S_{J_i} dem Zeitpunkt, zu dem das FTF in der Quelle ankommt.

Diese Art der Einlastung setzt das geplante Produktionsprogramm mit der Losgröße eins um. Außerdem erfolgt die feste Verknüpfung von Produktionsauftrag, WT und FTF für die abzuarbeitenden Produktionsprozesse des Produktes, was ebenfalls als Annahme in Abschnitt 2.4.2 eingeführt wird. Die verfügbaren WTs und FTFs sind dem zweiten Prinzip der Einlastung entsprechend dauerhaft im Einsatz.

Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement

Ist einem Fahrzeug einmal ein Produkt zugewiesen, also nach der Auftragseinlastung, trägt das FTF das Produkt nach Abschnitt 2.4.2 durch die notwendigen Prozesse in der MP. Dabei werden ereignisorientiert Action-Anfragen an den PPO-Agenten gestellt: nach der erfolgreichen Einlastung an der Quelle, nach jedem Produktionsschritt sowie an der Senke bei Fertigstellung. Über die Entscheidungen, an welche Station der Agent ein Produkt auf seinem Fahrzeug Schritt für Schritt leitet, löst der PPO-Agent das Auftragsrouting direkt und das Stationsmanagement implizit, indem er die Stationen für die Produktionsschritte auswählt. Die validen Actions pro Produktionsschritt der Produkte kennt der Agent hierbei. Dafür nutzt er das ihm zur Verfügung gestellte Produktionswissen. Außerdem sind die validen Actions im State codiert. Das Flottenmanagement erfolgt durch die direkte Koordination der Fahrzeuge. Störungen berücksichtigt der Agent in seinen Entscheidungen mithilfe der

State-Informationen, die ihm die Umgebung zur Verfügung stellt. Über die Reward-Signale lernt der Agent, wie er auf verschiedene Situationen optimal reagiert, um den Durchsatz zu maximieren.

5.4 Zusammenfassung und Einordnung des Ansatzes

Abschnitt 5.1 legt den grundlegenden Stand der Forschung lernender Steuerungsansätze dar, auf dessen Grundlage in Abschnitt 5.1.3 ein Grobkonzept abgeleitet wird. Anschließend beschreibt Abschnitt 5.2 die Grundlagen zur DRL-Methode AC-PPO, um die Entscheidungskategorien der MP zu lösen. Abschnitt 5.3 nutzt diese Methodik zur Beschreibung des Feinkonzepts des lernenden PPO-Agenten, wobei der Agent die Entscheidungen mittels „trial-and-error“ lernt und anschließend das Gelernte anwendet. Die folgenden Stichpunkte ordnen den lernenden Ansatz in die in Abschnitt 2.4.2 gestellten Anforderungen an eine MP-Steuerung ein. Die einzelnen Punkte beschreiben hierbei entweder, wie die bereits beschriebenen Inhalte die Anforderungen adressieren, oder wie sie in der späteren Implementierung in Abschnitt 6.2.4 umzusetzen sind:

- **Auftragseinlastung**: Die Auftragseinlastung orientiert sich am Conwip-Prinzip. Abschnitt 5.3.2 beschreibt die konzeptionellen Zusammenhänge. Der Zeitpunkt der Einlastung ergibt sich aus der Entscheidung des PPO-Agenten ein FTF zur Quelle zur Abholung eines bestimmten Produkttyps zu schicken. Außerdem weist der PPO-Agent damit einem einzulastenden Produkt ein FTF und einen WT zu, welche das Produkt für sämtliche Produktionsoperationen begleiten.

- **Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement**: Die an die Einlastung anschließenden Entscheidungen zu Stationsmanagement, Auftragsrouting, Flottenmanagement und Störungsmanagement sind Ergebnis der für die Fahrzeuge getroffenen Entscheidungen des PPO-Agenten. Dabei entscheidet der Agent Produktionsschritt für Produktionsschritt, wohin er das Produkt mithilfe eines Fahrzeugs als Nächstes schickt. Informationen zu Stationsbelegungen und Störungen findet der Agent hierbei im ihm zur Verfügung gestellten State-Vektor. Die Inhalte hierzu finden sich in Abschnitt 5.3.1 und Abschnitt 5.3.2.

- **Durchsatzmaximierung**: Die Durchsatzmaximierung wird durch die Gestaltung der Reward-Funktion des Agenten in Abschnitt 5.3.1 erreicht, welche auf eine Durchlaufzeitminimierung ausgelegt ist, was wiederum nach dem Gesetz von Little zum Durchsatzmaximum führt.

- **Echtzeitfähigkeit**: Echtzeitfähigkeit ermöglicht der Offline-Lernprozess, wonach im Online-Steuerungsfall lediglich echtzeitfähige Anfragen an den PPO-Agenten notwendig sind.

- **Datenverfügbarkeit**: Da der Agent zentral agiert, braucht eine reale Online-Steuerung eine zentrale Datenablage, welche den State des Produktionssystems zur Verfügung stellt. Die Daten sind hierfür von den jeweiligen Produktionsentitäten zu kommunizieren. Dass dies aus zentraler Sicht möglich ist, zeigt die Implementierung des in Kapitel 3 entwickelten zentralen Steuerungsansatzes. Im Gegensatz dazu steht die Datenverfügbarkeit im Sinne der Anforderung aus Abschnitt 2.4.2 im lernenden Ansatz nicht im Fokus. Wichtiger ist zuerst eine Minimierung der Rechenzeit in der Offline-Lernphase, um schnellstmöglich Lernerfolge zu erzielen. Dabei wirkt die Implementierung auf verteilten Recheneinheiten mit Fokus auf die Datenverfügbarkeit limitierend. Die auf Lerngeschwindigkeit ausgerichtete Implementierung beschreibt Abschnitt 6.2.4.

- **Generalisierbarkeit**: Generalisierbarkeit auf unterschiedliche Produktionskonfigurationen im Rahmen der in Abschnitt 2.4.2 definierten Systemgrenze zeigt später Abschnitt 6.5. Folgende Punkte des Steuerungskonzepts ermöglichen derartige Generalisierungsexperimente: Einerseits ist die Auftragseinlastung flexibel, weil sie sich auf einen gegebenen Produktmix einstellt. Andererseits passt sich der State-Vektor auf die gegebene Anzahl an Stationen an und skaliert dabei linear. Produktionswissen über zu fertigende Produkttypen kann ebenfalls beim Agenten hinterlegt werden. Darauf basierend weiß der Agent, welche Actions für welches Produkt in welchem State valide sind.

Offen bleiben die Fragen zur Einstellung der AC-PPO-Parameter und zur Implementierung. Da der Parameterraum bei AC-PPO zu groß ist, um alle Parameter einzeln zu untersuchen, orientiert sich die in der Evaluation in Kapitel 6 verwendete Konfiguration an PPO-Veröffentlichungen. Außerdem untersucht Abschnitt 6.3.3 den Einfluss des Action Maskings sowie den Einfluss der Anzahl paralleler Umgebungen, mit denen der Agent interagiert. Das Implementierungskonzept als zweite offene Frage ist Thema in Abschnitt 6.2.4.

Der entwickelte lernende Ansatz deckt die dritte der drei Steuerungsarchitekturen ab, welche die vorliegende Arbeit betrachtet. Damit komplettiert dieses Kapitel den Kern der Arbeit.

6 Simulationsbasierte Evaluation der Steuerungsalgorithmen

Mithilfe einer Evaluation mittels VIBN beantwortet dieses Kapitel die dritte Forschungsfrage aus Abschnitt 1.2, welche Steuerungsarchitektur unter welchen Rahmenbedingungen die besten Ergebnisse für die MP erzielt. Zuerst leitet Abschnitt 6.1 ein Evaluationsbeispiel aus einer E-Achsproduktion der Automobilindustrie ab. Abschnitt 6.2 beschreibt in der Folge die Umsetzung der Algorithmen im Framework der VIBN. Abschnitt 6.3 geht anschließend auf ansatzspezifische Untersuchungen und Parametereinstellungen ein, bevor Abschnitt 6.4 die Optimalität der Ansätze im statischen Produktionsfall (ohne Stationsstörungen) und die Robustheit gegenüber Störungen im dynamischen Produktionsumfeld (mit Stationsstörungen) untersucht. Abschnitt 6.5 leitet danach erweiterte Produktionsszenarien ab und prüft die Steuerungsansätze auf Generalisierbarkeit in Bezug auf die Produktionskonfiguration. Abschließend fasst Abschnitt 6.6 die Ergebnisse zusammen und schlussfolgert Handlungsempfehlungen für die Umsetzung von MP-Steuerungen. Für die vergleichenden Untersuchungen erfolgt ein Performanzabgleich mit einer Heuristik, die sich für jeden Produktionsschritt anhand der kürzesten Warteschlange entscheidet, sowie gegen ein theoretisches Durchsatzoptimum für die jeweilige Produktionskonfiguration. Die erstellten Diagramme basieren auf den Python-Bibliotheken [306] Matplotlib [307] und Seaborn [308].

6.1 Evaluationsbeispiel aus der Automobilindustrie

Zur Evaluation der zentralen, dezentralen und lernenden Steuerung nutzt diese Arbeit ein reales Produktionsbeispiel aus der Automobilindustrie, welches den Problemkontext aus Abschnitt 2.4.2 repräsentiert. In Anlehnung an eine E-Achsproduktion handelt es sich um eine Produktionshalle mit $n_M = 25$ Stationen. Abbildung 6.1 zeigt die abstrahierte matrixförmige Anordnung der Stationen. Zur einfacheren Darstellung erhält eine Station $M_{m,n}$ eine Matrixindizierung nach Reihen m und Spalten n. Die Fahrzeuge bewegen sich auf einem zweispurigen Wegenetz. Abhängig von der Fahrzeuggeschwindigkeit v_V ergeben sich die Transportzeiten aus den Wegstrecken, wobei ein Fahrzeug mittels A*-Algorithmus den kürzesten Weg sucht. Die Produktionsaufträge starten in der Quelle und enden in der Senke.

Die dargestellte MP fertigt primär zwei Produkte PT_1 und PT_2 (Evaluationsbeispiel 1).

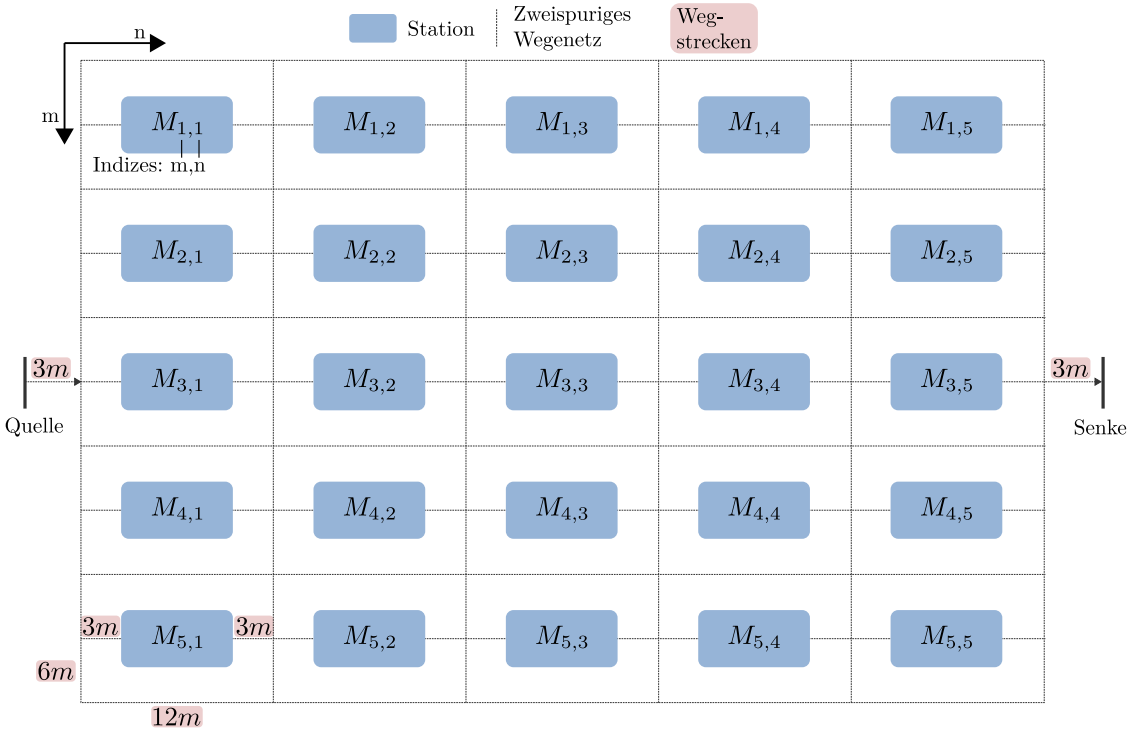


Abbildung 6.1: Matrixanordnung von $n_M = 25$ Stationen angelehnt an eine E-Achsproduktion aus der Automobilindustrie, Quelle: Eigene Darstellung in Anlehnung an [34].

Der dritte Produkttyp PT_3 erweitert den Produktmix für spätere Simulationen hinsichtlich der Generalisierbarkeit der Steuerungen (Evaluationsbeispiel 2). Tabelle 6.1 zeigt die Prozessschritte O_{ipl} pro Produkttyp sowie ihre Prozesszeiten t_{ipl} und die für O_{ipl} alternativ zur Verfügung stehenden Stationen $M_{ipl} \subseteq M$. Letzteres impliziert, dass eine Station $M_{n,m} \in M_{ipl}$ die zur Abarbeitung von Operation O_{ipl} notwendigen Fähigkeiten F_{ipl} bietet.

Unter der realitätsbasierten Annahme eines 1:1-Produktmixes lässt sich für die gegebene Konfiguration aus zwei Produkttypen PT_1 und PT_2 ein theoretisches Durchsatzoptimum D_{max} als obere Grenze des Systems ableiten. Die obere Grenze orientiert sich an den Engpassprozessen bzw. -stationen der MP. Bei einem 1:1-Produktmix ist der Engpass hierbei, nach einer Analyse der Produkt- und Stationskonfiguration, der ersten Operation der beiden Produkttypen und den zugehörigen Stationen zuzuordnen. D_{max} berechnet sich zu:

$$D_{max} = \frac{3600\frac{s}{h}}{\frac{177s+204s}{2\cdot 2}} = 37.8\frac{1}{h}, \tag{6.1}$$

wobei sich die Prozesszeiten von $177s$ und $204s$ auf zwei Produkte (PT_1, PT_2) und zwei Stationen ($M_{2,1}$, $M_{3,1}$) verteilen. Umgelegt auf eine Stunde ergibt sich ein theoretisch maximaler Durchsatz von 37.8 Produkten pro Stunde für eine theoretische Zykluszeit von $t_{\text{zyk}} = \frac{177s+204s}{2\cdot 2} = 95.25s$ zwischen der Fertigstellung zweier Produkte.

Darauf basierend lässt sich nach dem Gesetz von Little in (2.5) ein theoretisches FTF/WT-Minimum als untere Grenze ableiten. Dieses beantwortet die Frage, wie viele Fahrzeuge bzw.

Tabelle 6.1: Zu produzierende Produkttypen PT_1 und PT_2 in Evaluationsbeispiel 1 sowie Zusatzprodukt PT_3 für Evaluationsbeispiel 2.

	Produkttyp PT_1		**Produkttyp** PT_2		**Produkttyp** PT_3	
Nr.	t_{ipl} [s]	$M_{ipl} \subseteq M$	t_{ipl} [s]	$M_{ipl} \subseteq M$	t_{ipl} [s]	$M_{ipl} \subseteq M$
1	177	$\{M_{2,1}, M_{3,1}\}$	204	$\{M_{2,1}, M_{3,1}\}$	204	$\{M_{2,1}\}$
2	120	$\{M_{4,1}, M_{5,1}\}$	120	$\{M_{4,1}, M_{5,1}\}$	120	$\{M_{4,1}\}$
3	190	$\{M_{4,2}, M_{5,2}\}$	190	$\{M_{4,2}, M_{5,2}\}$	190	$\{M_{4,2}\}$
4	60	$\{M_{2,2}, M_{3,2}\}$	60	$\{M_{2,2}, M_{3,2}\}$	60	$\{M_{2,2}, M_{3,2}\}$
5	140	$\{M_{2,3}, M_{3,3}\}$	140	$\{M_{2,3}, M_{3,3}\}$	140	$\{M_{2,3}, M_{3,3}\}$
6	49	$\{M_{2,4}, M_{3,4}\}$	49	$\{M_{2,4}, M_{3,4}\}$	49	$\{M_{2,4}, M_{3,4}\}$
7	39	$\{M_{4,3}, M_{5,3}\}$	39	$\{M_{4,3}, M_{5,3}\}$	39	$\{M_{4,3}, M_{5,3}\}$
8	40	$\{M_{3,5}\}$	82	$\{M_{4,4}, M_{5,4}\}$	82	$\{M_{4,4}, M_{5,4}\}$
9	122	$\{M_{4,4}, M_{5,4}\}$	178	$\{M_{4,5}, M_{5,5}\}$	178	$\{M_{4,5}, M_{5,5}\}$
10	137	$\{M_{4,5}, M_{5,5}\}$	40	$\{M_{3,5}\}$	40	$\{M_{3,5}\}$
11	44	$\{M_{3,5}\}$	80	$\{M_{1,5}, M_{2,5}\}$	80	$\{M_{1,5}, M_{2,5}\}$
12	155	$\{M_{1,5}, M_{2,5}\}$	72	$\{M_{2,4}, M_{3,4}\}$	72	$\{M_{2,4}, M_{3,4}\}$
13	72	$\{M_{2,4}, M_{3,4}\}$	37	$\{M_{3,5}\}$	37	$\{M_{3,5}\}$
14	0	$\{Senke\}$	0	$\{Senke\}$	0	$\{Senke\}$

WTs ($n_V = n_{\mathrm{WT}}$) mindestens benötigt werden, um D_{max} erreichen zu können. Die Fahrzeuge und WTs bilden hierbei den Produktionsbestand PB. Unter den realitätsbasierten Annahmen einer Fahrzeuggeschwindigkeit von $v_V = 2\ m/s$ (Default), von Pufferkapazitäten in EP und AP von $n_{\mathrm{AP}} = n_{\mathrm{EP}} = 1$ (Default) sowie einer Transportzeit pro Pufferplatz von $t_{\mathrm{PP}} = 3s$ (Default) berechnet sich das Minimum $n_{V,min}$ ($= PB_{min}$) für die beiden Produkttypen im 1:1-Mix zu:

$$n_{V,min} \geq D_{max} \cdot DZ_{J,m,min} = \frac{37.8\frac{1}{h} \cdot 1628.5s}{3600\frac{s}{h}} = 17.1, \tag{6.2}$$

wobei $DZ_{J,m,min}$ die mittlere Durchlaufzeit über alle Routen aller Aufträge darstellt, berechnet unter der Annahme, dass keine Wartezeiten anfallen und keine Stationen ausfallen. In der Theorie sind in der angenommenen Konfiguration daher mindestens 18 Fahrzeuge zur Erreichung des Durchsatzoptimums erforderlich.

Diese beiden Werte sind später für die Auslegung und Auswertung der Experimente relevant. Tabelle 6.2 dokumentiert die aus der Realität abgeleitete Default-MP für die späteren Experimente. Dabei ist zu beachten, dass zur Produktion von PT_1 und PT_2 vorerst lediglich 21 von $n_M = 25$ Stationen aktiv sind, wie Tabelle 6.1 zeigt. Die weiteren vier Stationen

repräsentieren Spielraum für Änderungen in der MP-Konfiguration.

Tabelle 6.2: Default-Konfiguration der MP, auch bezeichnet als Evaluationsbeispiel 1.

Parameter	**Beschreibung**
$n_M = 25$	MP bestehend aus 25 Stationen in 5x5-Matrixanordnung
$n_V = n_{\mathrm{WT}} = 18$	18 Fahrzeuge und WTs zum Transport der Produkte
Produktmix 50:50 (1:1)	Zwei Produkttypen PT_1 und PT_2 à 50 Aufträgen im 1:1-Wechsel
$v_V = 2\ m/s$	Fahrzeuggeschwindigkeit von 2 m/s
$n_{\mathrm{AP}} = n_{\mathrm{EP}} = 1$	Pufferkapazitäten zu eins gesetzt
$t_{\mathrm{PP}} = 3s$	Fließbandbedingte Transportzeit pro Pufferplatz zu $3s$ gesetzt
$D_{max} = 37.8\ 1/h$	Maximaler Durchsatz von 37.8 Produkten pro Stunde
$t_{\mathrm{zyk}} = 95.25s$	Minimale Zykluszeit von $95.25s$

Abschließend fassen die folgenden Punkte die getroffenen Annahmen aus Abschnitt 2.4.2 im Hinblick auf die fünf Entscheidungskategorien zusammen und stellen damit nochmals den Untersuchungsrahmen der Evaluationsbeispiele klar:

- **Auftragseinlastung**: Die Auftragseinlastung folgt einem definierten Produktmix bzw. Produktionsprogramm. Die Zeitpunkte der Einlastung S_{J_i} eines Auftrags J_i legt die jeweilige Steuerung fest.

- **Stationsmanagement**: Jede Station $M_{m,n}$ besteht aus einem EP und einem AP, welche den Bearbeitungsplatz umrahmen. Die Puffer arbeiten nach dem FIFO-Prinzip. Der Stationseingang befindet sich auf der linken Seite der Station, der Stationsausgang auf der rechten Seite.

- **Auftragsrouting**: Alternative Routen eines Auftrags J_i ergeben sich aus der Operationsflexibilität, wobei eine Operation O_{ipl} an Stationen M_{ipl} bearbeitet werden kann.

- **Flottenmanagement**: Ein FTF V_j und ein WT WT_o begleiten ein Produkt nach Auftragseinlastung für die gesamten Abläufe bis zur Fertigstellung in der Senke. An einer Station wartet ein FTF in einer dafür vorgesehenen Parkzone, bis der nächste Produktionsschritt ansteht. Ist der EP der Station voll, wartet das Fahrzeug vor der Station auf dem Wegenetz. Überholvorgänge auf dem Wegenetz sind nicht möglich.

- **Störungsmanagement**: Ausschließlich Stationen fallen nach einer gegebenen $MTTR$ und $MTBF$ bzw. Verfügbarkeit V_M aus. Fahrzeuge sind zu 100% verfügbar.

Der nächste Abschnitt legt die Umsetzung des Evaluationsbeispiels und der Steuerungsalgorithmen mithilfe des Konzepts der VIBN dar.

6.2 Implementierung der Algorithmuskonzepte

Die drei vorgestellten Steuerungsansätze unterscheiden sich signifikant in der Entscheidungsfindung. Beim zentralen Ansatz berechnet eine zentrale Recheneinheit globale Ablaufpläne für das Produktionssystem. Beim dezentralen Ansatz verhandeln Produkte mit Stationen um die Ressourcenallokation. Der lernende Ansatz agiert mittels „trial-and-error“ und sammelt dadurch Erfahrungen, die zur Steuerung genutzt werden. Um die drei Steuerungsansätze trotz der spezifischen Eigenschaften unter gleichen Rahmenbedingungen vergleichen zu können, verwendet die vorliegende Arbeit das Konzept der VIBN aus Abschnitt 2.3.4, welches sich durch den Dreiklang aus Simulation, Schnittstelle und Steuerungslogik auszeichnet. Dabei ist die Umsetzung an das in [37] vorgestellte MIL-Framework angelehnt, weil es:

- hinsichtlich unterschiedlicher MP-Konfigurationen sowohl auf Steuerungs- als auch auf Simulationsseite konfigurierbar ist,
- Dynamik in Form von Stationsstörungen modellieren lässt,
- zentrale, dezentrale und lernende Ansätze ermöglicht,
- alle Entscheidungskategorien dieser Arbeit berücksichtigen kann,
- verlässliche und schnelle Datenflüsse zwischen Steuerungsentitäten via „Message Queuing Telemetry Transport“ (MQTT) bietet,
- eine performante und skalierbare Schnittstelle zwischen Steuerung und Simulation via TCP/IP mit sich bringt und
- durch die Berücksichtigung der realen Kommunikation eine realitätsnahe, prototypische Implementierung der Algorithmen ermöglicht.

Weitere Details zur Implementierungsgrundlage dokumentiert Anhang A.1. Das Ziel der Umsetzung ist, eine ähnlich realitätsnahe Evaluationsumgebung zu generieren, wie beispielsweise dokumentiert in den analysierten Steuerungsansätzen von [248], [256].

Zuerst geht der Abschnitt hierfür auf Simulation und Schnittstelle in Abschnitt 6.2.1 ein. Daraufhin folgen die Umsetzungen der drei Steuerungsalgorithmen in Abschnitt 6.2.2, Abschnitt 6.2.3 und Abschnitt 6.2.4. Abschließend dokumentiert Abschnitt 6.2.5 Plausibilitätstests, die zur Validierung und Verifikation der Implementierung Anwendung finden.

6.2.1 Modulare Produktion und Schnittstelle

Simulation und Schnittstelle sind zwei essenzielle Bestandteile der VIBN. Die Simulation bildet die Produktionsabläufe der MP ab. Die Schnittstelle verbindet MP und Steuerung.

Die Simulation der MP aus Abschnitt 6.1 erfolgt hierbei mittels der DES-Software Tecnomatix Plant Simulation in der Version 15.0, einem Tool zur Materialflusssimulation, welches auch Schnittstellenpakete anbietet, um Kommunikation mit einer Steuerungslogik zu ermöglichen [309, S. 661ff.]. Tecnomatix Plant Simulation findet hierbei auch in der analysierten Literatur zur Modellierung von Produktionssystemen erfolgreich Anwendung [269], [284].

Initialphase der Simulation und Auftragseinlastung

In der Initialphase der Simulation werden die n_J zu fertigenden Produktionsaufträge, die n_{WT} zur Verfügung stehenden WTs sowie die n_V Fahrzeuge in der Quelle erzeugt, dargestellt in Abbildung 6.2. Die Aufträge liegen dabei von Anfang an bereit. Sie sind in Abhängigkeit der WT- und Fahrzeugverfügbarkeit zu starten. Dabei gilt es, die initialisierten Reihenfolgen der Fahrzeuge und WTs einzuhalten, beginnend mit V_1 und WT_1.

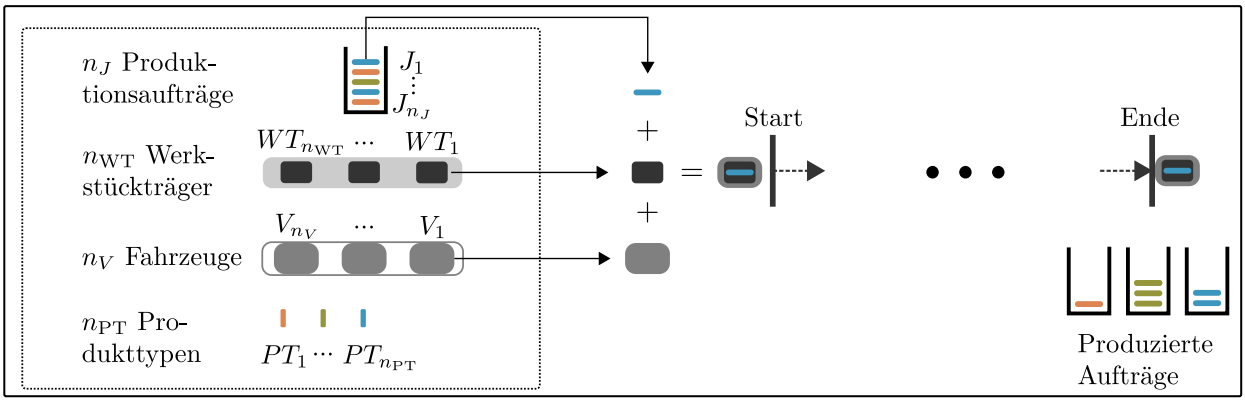


Abbildung 6.2: Darstellung der Quelle der MP und der damit verbundenen Auftragseinlastung bzw. initialen Simulationsphase.

Dann startet die Simulation mit den ersten Produktionsereignissen, die vor allem die Auftragseinlastung betreffen. Die Auftragseinlastung hängt hierbei vom Steuerungsansatz ab, beschrieben in Kapitel 3, Kapitel 4 und Kapitel 5. Im zentralen Ansatz wird der erste Auftrag eingelastet und deshalb die erste Schedule-Berechnung angestoßen, welche das erste Stationsziel vorgibt. Die weiteren Aufträge folgen jeweils mit dem einzustellenden Abstand ΔT_{AE}. Im dezentralen Ansatz erfolgt die erste Auktion zur Bestimmung der Station für den ersten Produktionsschritt des einzulastenden Auftrags. Die weiteren Aufträge folgen direkt im Anschluss, aber nacheinander. Im lernenden Ansatz wird eine erste Action für Fahrzeug V_1 angefragt, um einen Auftrag zu starten und ihn dann anschließend mithilfe einer zweiten Action zur ersten Station zu führen. Auch dieser Vorgang wiederholt sich für die anderen Fahrzeuge, um weitere Aufträge zu starten. Die Zuweisung des Produkttypen bei der jeweiligen Einlastung obliegt dem Steuerungsansatz.

Simulierte Produktionsabläufe der MP

Die ereignisorientierte Simulation arbeitet die in der Produktion anstehenden Ereignisse, welche u.a. durch die Entscheidungen der Steuerungslogik entstehen, Ereignis für Ereignis ab. Auf die Initialisierungsphase folgen dabei die Produktionsabläufe in der MP. Abbildung 6.3 fasst die Umsetzung der Simulation schematisch zusammen. Produktionsaufträge starten nach obigem Prinzip in der Quelle und bewegen sich dann Entscheidung für Entscheidung durch die Produktion. Beim Durchlauf durch die MP können sie sechs unterschiedliche Status annehmen:

- **Quelle**: Der Auftrag befindet sich in der Quelle.
- **FTF**: Der Auftrag befindet sich auf einem FTF und wird zu einer Station (oder zur Senke) mit Transportzeit $t_{\mathrm{TR},ipl}$ transportiert. Bei voller Station wartet der Auftrag vor der Station, wodurch sich $t_{\mathrm{TR},ipl}$ verlängert.
- **EP**: Der Auftrag befindet sich im EP einer Station und wird mit Transportzeit $t_{\mathrm{EP},ipl}$ zur Bearbeitung transportiert. Bei belegtem Bearbeitungsplatz wartet der Auftrag vor der Bearbeitung, wodurch sich $t_{\mathrm{EP},ipl}$ verlängert.
- **IB**: Der Auftrag befindet sich in Bearbeitung an einer Station mit Bearbeitungszeit t_{ipl}. Ist eine Station gestört, pausiert sie die Bearbeitung am aktuell geladenen Produkt. Nach Störungsende setzt sie die Bearbeitung mit der verbleibenden Prozesszeit fort.
- **AP**: Der Auftrag befindet sich im AP einer Station und wird zum Fahrzeug mit Transportzeit $t_{\mathrm{AP},ipl}$ gefördert. Bei fehlender Erlaubnis zur Weiterfahrt wartet der Auftrag im Puffer, wodurch sich $t_{\mathrm{AP},ipl}$ verlängert.
- **Senke**: Der Auftrag wurde in der Senke nach Fertigstellung abgeliefert.

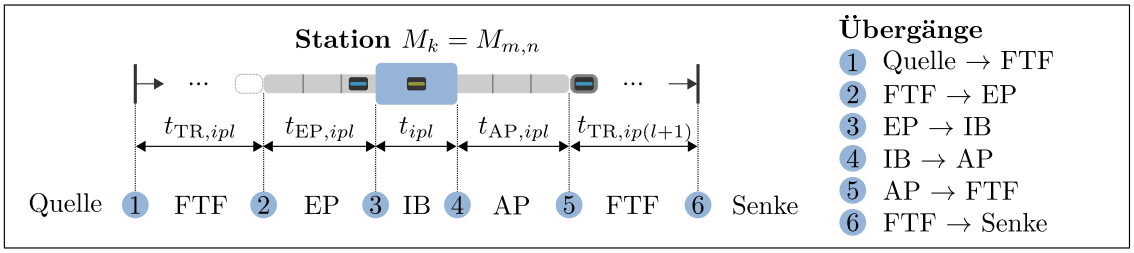


Abbildung 6.3: Statusübergänge eines Produktionsauftrags in der MP mit den zugehörigen Prozesszeiten.

Insgesamt ergeben sich daraus sechs Statusübergänge, die die Abbildung ebenfalls zeigt. Die ablaufenden Zeiten pro Status sind grundsätzlich deterministisch. $t_{\mathrm{TR},ipl}$ ergibt sich aus

der zurückgelegten Entfernung mit Geschwindigkeit v_V. $t_{\mathrm{EP},ipl}$ entspricht der fließbandbedingten Mindest-Transportzeit $t_{\mathrm{EP},min}$ im Puffer. t_{ipl} entspricht einer definierten Bearbeitungszeit. $t_{\mathrm{AP},ipl}$ entspricht ebenfalls der Mindest-Transportzeit $t_{\mathrm{AP},min}$ im Puffer, wobei gilt: $t_{\mathrm{EP},min} = t_{\mathrm{AP},min}$. Die deterministischen Zeiten verlängern sich durch die in der Aufzählung beschriebenen potenziell auftretenden Wartezeiten. Dynamik resultiert damit aus der gegenseitigen Beeinflussung der Aufträge sowie aus Stationsstörungen ($MTTR$, $MTBF$, V_M). Zeiten für die WT/Produkt-Übergabe von einem FTF an eine Station oder umgekehrt werden vernachlässigt. Sie verlängern lediglich die fließbandbedingten Transportzeiten. Es gilt damit die Annahme, dass der Fließbandtransport die Übergabezeiten mit einschließt. Gleiches gilt für Übergabezeiten in Quelle und Senke.

Fertigstellung von Aufträgen in der Simulation

Fahrzeuge und WTs, die Aufträge in der Senke fertigstellen, sendet die Simulation direkt zur Quelle zurück, um neue Aufträge zu starten. Als Sonderfall gilt hier der lernende Ansatz aufgrund der Konzeptionierung der Auftragseinlastung. Der PPO-Agent entscheidet bereits an der Senke, welcher Auftrag in der Quelle einzulasten ist, damit dem Fahrzeug bei Ankunft in der Quelle diese Information zur Verfügung steht. Trotzdem resultiert aus dieser Entscheidung in der Senke immer die Fahrt zur Quelle.

Simulation von Störungen

Zur Simulation von Stationsstörungen wählt Tecnomatix Plant Simulation standardmäßig eine Erlang-Verteilung unter Berücksichtigung der $MTTR$ und der Verfügbarkeit V_M. Aus der Erlang-Verteilung bezieht Plant Simulation die jeweilige Störungsdauer, wobei eine untere und obere Grenze angegeben werden kann, die nicht unter- bzw. überschritten wird. Die $MTBF$ berechnet sich die Simulation aus $MTTR$ und V_M. Für die tatsächlichen Abstände zwischen zwei Störungen auf Basis der $MTBF$ nutzt Plant Simulation eine negative Exponentialverteilung. Die Abstände beziehen sich dabei auf die Einsatzzeit einer Station. Die notwendigen statistischen Berechnungen führt Plant Simulation automatisiert durch. Jeder Station wird zu Beginn der Simulation ein eigener Zufallszahlenwert zugeordnet, wodurch sich die Störungen an den Stationen unterscheiden. Durch Variation dieser Werte können gezielt verschiedene Störszenarien generiert und wiederholt werden, wobei die spätere Evaluation von unterschiedlichen Störungs-Seeds spricht.

Schnittstelle zur Steuerungslogik

Die in Abbildung 6.3 dargestellten Statusübergänge sind an die Steuerungslogik zu kommunizieren, weil diese ein aktuelles Gesamtbild zur MP braucht, um valide Entscheidungen zu treffen. Zusätzlich zu den Statusübergängen fließen weitere Informationen an die Steuerung.

Die hierfür notwendige Schnittstelle wird durch eine TCP/IP-Verbindung abgebildet, welche erstens im Schnittstellenpaket von Tecnomatix Plant Simulation enthalten ist und zweitens eine performante Kommunikation sicherstellt.

Den Informationsfluss zu und von der Steuerungslogik regen entweder zeitliche Trigger oder ereignisorientierte Trigger an. Abbildung 6.4 gibt eine Übersicht zu den implementierten Verbindungen, wobei je nach Steuerungsansatz unterschiedliche Verbindungen aktiv sind. Die Pfeile stellen teilweise einseitige und teilweise beidseitige Informationsflüsse dar.

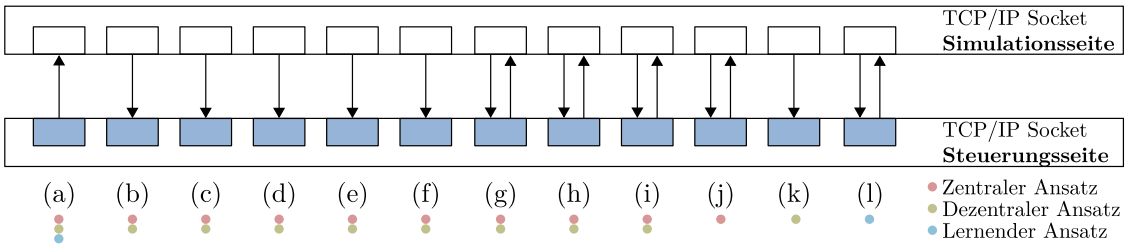


Abbildung 6.4: Socket-Verbindungen als Schnittstelle vom Simulationsmodell zur Steuerungslogik, wobei die drei Algorithmuskonzepte unterschiedliche TCP/IP-Verbindungen nutzen.

Die Socket-Verbindungen ermöglichen die Konfiguration der Simulation, die Aktualisierung der Wissensbasis der Steuerungslogik, das Abrufen von Steuerungsentscheidungen sowie das Anregen neuer Steuerungsberechnungen. Die folgende Aufzählung beschreibt die ansatzspezifisch kommunizierten Informationen sowie ihre Trigger:

- (a) **Konfiguration**: Vor Simulationsstart wird das Produktionssystem konfiguriert. Dabei erfolgt beispielsweise die Einstellung der Anzahl der Stationen n_M und deren Matrixanordnung (z.B. 5x5), der Anzahl der Fahrzeuge n_V und ihrer Geschwindigkeit v_V, der Anzahl der WTs n_{WT}, der Störungsparameter etc.

- (b) **Update globale Zeit**: Jede Simulationssekunde überträgt die Simulation die aktuelle Simulationszeit an die Steuerung, um Steuerung und Simulation zu synchronisieren.

- (c) **Update Stationswissen**: Das aktuelle Wissen einer Station wird ereignisorientiert aktualisiert, wenn sie von einem der Statusübergänge aus Abbildung 6.3 betroffen ist.

- (d) **Update FTF-Wissen**: Das aktuelle Wissen eines FTFs wird ereignisorientiert aktualisiert, wenn es von einem der Statusübergänge aus Abbildung 6.3 betroffen ist.

- (e) **Update WT/Produkt-Wissen**: Das aktuelle Wissen eines WTs bzw. zugewiesenen Produkts wird ereignisorientiert aktualisiert, wenn es von einem der Statusübergänge aus Abbildung 6.3 betroffen ist.

- (f) **Update Störungen**: Bei Störungsstart $t_{\mathrm{stoer,start}}$ und Störungsende $t_{\mathrm{stoer,ende}}$ informiert die Simulation die Steuerungslogik über dieses Ereignis.

- (g) **Anfrage neuer Auftrag für WT**: Sind ein zu startendes Produkt, ein WT und ein FTF in der Quelle verfügbar, sendet die Simulation die Information über dieses Ereignis an die Steuerung, um einen Auftrag zuzuweisen.
- (h) **Anfrage neues Ziel für WT/Produkt**: Bei den Statusübergängen eins und fünf aus Abbildung 6.3 fragt der WT bzw. das Produkt ereignisorientiert das Stationsziel für den nächsten Produktionsschritt an und folgt diesem Ziel.
- (i) **Anfrage neues Ziel für FTF**: Bei den Statusübergängen eins und fünf aus Abbildung 6.3 fragt das FTF ereignisorientiert das Stationsziel für den nächsten Produktionsschritt des getragenen Produkts an und folgt diesem Ziel.
- (j) **Anfrage Schedule-Berechnung**: Bei Auftragseinlastung zum Zeitpunkt S_{J_i}, zeitlich periodisch mit Abstand Δt_s sowie bei Störungsstart $t_{\text{stoer,start}}$ und Störungsende $t_{\text{stoer,ende}}$ an einer Station löst die Simulation eine Schedule-Berechnung vom zentralen Steuerungsansatz aus.
- (k) **Anfrage Prüfung, ob Auktionsstart**: Alle fünf Sekunden sendet die Simulation ein Signal an die dezentrale Steuerungslogik, damit die Stationen prüfen, ob sie freie Kapazität in einer Auktion zur Verfügung stellen möchten.
- (l) **Anfrage Action für FTF**: Im lernenden Ansatz initiiert das Ereignis eines Fahrzeugs, keinen neuen Fahrauftrag zu haben, eine Action-Anfrage an den PPO-Agenten, um diese Entscheidung zu treffen (vgl. Abbildung 5.6).

Wie Abbildung 6.4 zeigt, unterscheiden sich der zentrale und dezentrale Ansatz lediglich bei den Sockets (j) und (k), die jeweils die ansatzspezifische Entscheidungsfindung betreffen. Der lernende Ansatz nutzt hingegen lediglich den Konfigurations-Socket (a) und die RL-spezifische Action-Anfrage über Socket (l). Dabei ist der lernende Ansatz auf die Performanz zum Lernen in der Lernphase ausgerichtet, wo eine schnelle und unkomplizierte Kommunikation wesentlich ist. Während der zentrale und dezentrale Ansatz auch die Anforderung an Datenverfügbarkeit durch Kommunikation zwischen unterschiedlichen Recheneinheiten (vgl. Abschnitt 2.4.2) umsetzen, fokussiert sich der lernende Ansatz also auf Recheneffizienz beim Lernen. Dennoch sind die Simulationsmodelle, an die die ansatzspezifische Schnittstelle angedockt ist, identisch. Dies stellt, bis auf die Kommunikationsunterschiede, identische Rahmenbedingungen zum Vergleich der Ansätze sicher. Auf die ansatzspezifischen Unterschiede in der Steuerungslogik gehen die folgenden drei Abschnitte ein.

6.2.2 Rollierendes Scheduling

In einer realen Produktion existieren Produktionsentitäten zugeordnete Recheneinheiten, die in die Produktionssteuerung zu integrieren sind. Zur realitätsnahen Implementierung und

Erfüllung der Anforderung an Datenverfügbarkeit aus Abschnitt 2.4.2 erfolgt die Umsetzung der zentralen Steuerungslogik deshalb in einem virtuellen Rechencluster mittels Virtueller Maschinen (VM). Die auf die VMs verteilten Steuerungs- und Verwaltungsfunktionen wurden hierzu mithilfe der visuellen Programmierumgebung Node-RED[1] programmiert. Die Node-RED-basierten VMs setzen die Funktionen in folgender Konfiguration um:

- 1x VM für zentral **koordinative Aufgaben**: TCP/IP-Nachrichten-Broker, Verteilung der Nachrichten auf Steuerungsseite per MQTT, Koordination der Auftragseinlastung.
- 1x VM für die zentrale **Scheduling-Einheit**: Verwaltung der zentralen Wissensbasis, Anstoßen der Schedule-Berechnung nach Abschnitt 3.3.2 mittels Python und der GA-Bibliothek DEAP [312], Verteilung der Entscheidungen an die Produktionsentitäten.
- 4x VM mit je 5 Docker-Containern [313] für die Verwaltung der **Stationen**.
- 4x VM mit je 5 Docker-Containern für die Verwaltung der **Fahrzeuge**.
- 4x VM mit je 5 Docker-Containern für die Verwaltung der **WTs/Produkte**.

Je nach Produktionskonfiguration wird jeder Station, jedem Fahrzeug und jedem WT bzw. Produkt ein eigener Docker-Container zugewiesen. Überschreitet die Anzahl der jeweiligen Produktionsentität die Anzahl der zur Verfügung stehenden Docker-Container, erfolgt eine automatisierte Verteilung der überschüssigen Entitäten auf die Container. In jedem Fall existiert für jede der Produktionsentitäten eine eigenständige Node-RED-Instanz. Jede VM nutzt hierbei einen zugewiesenen Arbeitsspeicher von 4GB sowie vier virtuelle Prozesskerne. Nur die zentrale Scheduling-Einheit arbeitet mit 16 GB Arbeitsspeicher. Das Simulationsmodell in Tecnomatix Plant Simulation läuft hierbei auf einer eigenen VM mit acht virtuellen Prozesskernen und 16GB Arbeitsspeicher. Die virtuellen Prozesskerne werden dynamisch zugewiesen und arbeiten mit Taktfrequenzen zwischen 2.27 und 2.8 GHz.

Obige Konfiguration setzt den in Kapitel 3 entwickelten zentralen Steuerungsansatz um. Abbildung 6.5 zeigt die abstrakten Zusammenhänge. Jegliche Kommunikation von der Simulation zur Steuerungslogik läuft über die eingeführten Sockets, welche auf Steuerungsseite von der VM für koordinative Aufgaben verwaltet werden. Die koordinative VM verteilt die Nachrichten per MQTT [314] mittels Publish-Subscribe an die von der Simulation zu adressierenden Node-RED-Instanzen. Hierbei enthalten die gesendeten Nachrichten entsprechende Codes, z.B. einen Stationsnamen, damit die VMs identifizieren können, ob die Nachricht für sie bestimmt ist oder sie ignoriert werden kann. Die direkte Kommunikation zwischen Node-RED-Instanzen, unabhängig von der koordinativen VM, nutzt ebenfalls Publish-Subscribe über MQTT.

[1] Verfügbar unter: https://nodered.org/ (besucht am 01.05.2022), Nachrichten im JSON-Format, Vernetzung von Hardware-Komponenten, visuelle Programmierung, gängig zur Entwicklung von IdD-Prototypen, z.B. in [310], [311].

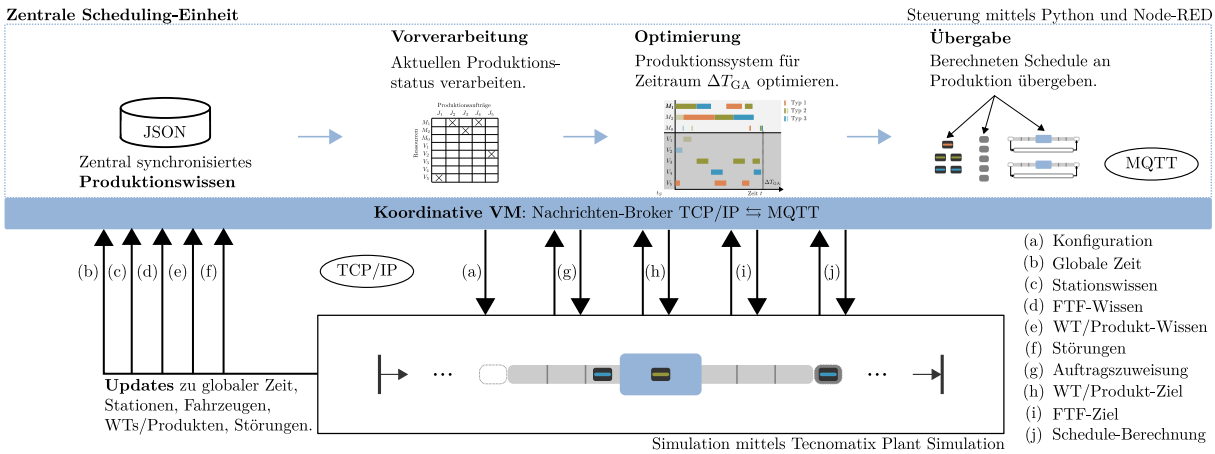


Abbildung 6.5: Umsetzung des zentralen Steuerungsansatzes.

Vor Simulationsstart konfiguriert die Steuerung das Simulationsmodell mittels Socket (a). Über die Sockets (b), (c), (d), (e) und (f) informiert die Simulation die zentrale Scheduling-Einheit über Zustandsänderungen in der Simulation, um die „JavaScript Object Notation“ (JSON)-Wissensbasis aktuell zu halten. Beispielsweise findet für Stationen und WTs/Produkte ein zeitliches Update der individuellen Planung statt, falls ein Übergang auf einen Bearbeitungsplatz (Übergang 3 in Abbildung 6.3) erfolgt. Dadurch ergibt sich ein Abgleich der geschätzten sowie real eingetretenen Startzeiten. Die zentrale Wissensbasis kommt dann wiederum zum Einsatz, wenn die Simulation über Socket (j) eine Schedule-Berechnung startet und die Scheduling-Einheit einen Schedule für die MP bereitstellt. Während der Optimierung stoppt die Simulation. Nach der Berechnung wird die Simulation fortgesetzt, wobei die getroffenen Entscheidungen mit Verzögerung von der Rechenzeit der Optimierung t_{opt} eingespielt werden, um Echtzeit zu simulieren. Während der Berechnung blockiert die Simulation die Statusübergänge aus Abbildung 6.3, damit die nach der Berechnung eingespielten Entscheidungen nicht veraltet sind. Startet die Simulation in dieser Zeit bereits eine neue Optimierung, wartet sie, bis t_{opt} abgelaufen ist, bevor der neue Optimierungslauf beginnt. Durch das Start-Stopp-Prinzip lässt sich die Simulation beschleunigen, um schneller Simulationsergebnisse zu erhalten.[2] Gleichzeitig bleibt die Realitätsnähe durch die nachträgliche Berücksichtigung der Berechnungszeit erhalten, wodurch Echtzeitfähigkeit nachgewiesen werden kann. Die zentrale Scheduling-Einheit verteilt die getroffenen Entscheidungen via MQTT an die betroffenen Stationen, Fahrzeuge und WTs/Produkte. Benötigt die Simulation Entscheidungen zu neuen Zielen für WT/Produkt und FTF (Sockets (h) und (i)), fragt sie diese bei den entsprechenden Node-RED-Instanzen der Produktionsentitäten an.

[2]Die Start-Stopp-Funktion lässt sich auch ausschalten. Dann wird der jeweilige Schedule parallel zur laufenden Simulation berechnet. Auch in diesem Fall dürfen aus den genannten Gründen während der Schedule-Berechnung keine Statusübergänge stattfinden. Ohne Start-Stopp-Funktion läuft die Simulation in Echtzeit und ist nicht beschleunigbar. Die Simulationsergebnisse sind jedoch identisch zur beschleunigten Variante, weshalb diese für die spätere Evaluation im Einsatz ist.

Benötigt ein WT einen neuen Produktionsauftrag, findet eine Abfrage über Socket (g) statt und die koordinative VM weist einen Auftrag sowie das FTF zu. Die jeweiligen Antworten der Node-RED-Instanzen werden wiederum über die koordinative VM und die zugeordnete Socket-Verbindung an die Simulation übertragen.

Das auf Steuerungsseite verwaltete zentrale Produktionswissen der **Scheduling-Einheit** betrifft alle Stationen, Fahrzeuge, WTs/Produkte, Störungen sowie GA-Parameter. Es gibt dem GA einen Überblick zum aktuellen Produktionsstatus als Input für die Optimierung. Die **koordinative VM** arbeitet hingegen mit Daten für die Auftragseinlastung. Anhang B.1 dokumentiert jeweils die zugehörigen Informationen, welche zur Umsetzung des in Abschnitt 3.3 entwickelten zentralen Steuerungskonzepts notwendig sind.

Die Node-RED-Instanzen der **Stationen, Fahrzeuge** und **WTs/Produkte** agieren demgegenüber passiv und werden von der aktiven zentralen Scheduling-Einheit sowie von der koordinativen VM informiert. Sie treffen keine Entscheidungen, speichern die getroffenen Entscheidungen jedoch ab, damit die Simulation die nächsten Schritte für WTs und Fahrzeuge direkt bei der zuständigen Node-RED-Instanz erfragen kann.

Dabei stellen Simulation, Schnittstelle und Steuerung die Aktualität aller Informationen sicher, wodurch die Anforderung an Datenverfügbarkeit erfüllt wird. Einige der Informationen, beispielsweise Stationsfähigkeiten und Transportzeiten, speichern die entsprechenden Node-RED-Instanzen bei der Initialisierung einmalig ab, da sie sich während der Simulation nicht verändern.

6.2.3 Auktionsbasiertes Multiagentensystem

In einer realen Produktion existieren Produktionsentitäten zugeordnete Recheneinheiten, die in die Produktionssteuerung zu integrieren sind. Zur realitätsnahen Implementierung und Erfüllung der Anforderung an Datenverfügbarkeit aus Abschnitt 2.4.2 erfolgt die Umsetzung der dezentralen Steuerungslogik ebenfalls in einem virtuellen Rechencluster mittels VMs mithilfe der visuellen Programmierumgebung Node-RED. Die Node-RED-basierten VMs setzen die Funktionen in folgender Konfiguration um:

- 1x VM für zentral **koordinative Aufgaben**: TCP/IP-Nachrichten-Broker, Verteilung der Nachrichten auf Steuerungsseite per MQTT, Koordination der Auftragseinlastung (**AE-Agent**).
- 4x VM mit je 5 Docker-Containern für **M-Agenten**: Verwaltung der Stationen, Durchführen von Auktionen zur Ressourcenallokation nach Abbildung 4.3.
- 4x VM mit je 5 Docker-Containern für **V-Agenten**: Verwaltung der Fahrzeuge.
- 4x VM mit je 5 Docker-Containern für **P-Agenten**: Verwaltung der WTs/Produkte, Teilnahme an Auktionen zur Ressourcenallokation.

Je nach Produktionskonfiguration wird jeder Station, jedem Fahrzeug und jedem WT bzw. Produkt ein eigener Docker-Container zugewiesen. Überschreitet die Anzahl der jeweiligen Produktionsentität die Anzahl der zur Verfügung stehenden Docker-Container, erfolgt wie bei der zentralen Umsetzung eine automatisierte Verteilung der überschüssigen Entitäten auf die Container. In jedem Fall existiert für jede der Produktionsentitäten eine eigenständige Node-RED-Instanz. Jede VM nutzt wie im zentralen Ansatz einen zugewiesenen Arbeitsspeicher von 4GB sowie vier virtuelle Prozesskerne. Das Simulationsmodell in Tecnomatix Plant Simulation läuft auf einer eigenen VM mit acht virtuellen Prozesskernen und 16GB Arbeitsspeicher. Die virtuellen Prozesskerne werden dynamisch zugewiesen und arbeiten mit Taktfrequenzen zwischen 2.27 und 2.8 GHz.

Obige Konfiguration setzt den in Kapitel 4 entwickelten dezentralen Steuerungsansatz um. Abbildung 6.6 zeigt die abstrakten Zusammenhänge. Jegliche Kommunikation von der Simulation zur Steuerungslogik läuft über die eingeführten Sockets, welche auf Steuerungsseite von der VM für koordinative Aufgaben verwaltet werden. Die koordinative VM verteilt die Nachrichten per MQTT [314] mittels Publish-Subscribe an die von der Simulation zu adressierenden Node-RED-Instanzen. Hierbei enthalten die gesendeten Nachrichten entsprechende Codes, z.B. einen Stationsnamen, damit die VMs identifizieren können, ob die Nachricht für sie bestimmt ist oder sie ignoriert werden kann. Die direkte Kommunikation zwischen Node-RED-Instanzen, unabhängig von der koordinativen VM, nutzt ebenfalls Publish-Subscribe über MQTT.

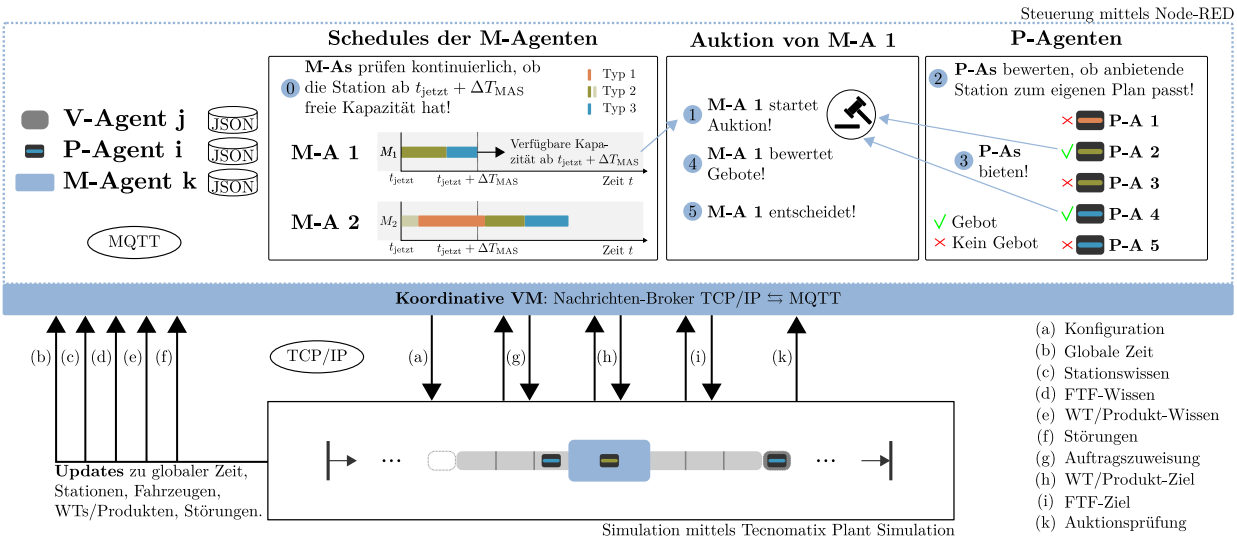


Abbildung 6.6: Umsetzung des dezentralen Steuerungsansatzes.

Vor Simulationsstart konfiguriert die Steuerung das Simulationsmodell mittels Socket (a). Über die Sockets (b), (c), (d), (e) und (f) informiert die Simulation die betroffenen M-, V- und P-Agenten über Zustandsänderungen in der Simulation, um die jeweilige JSON-Wissensbasis

aktuell zu halten. Beispielsweise findet für Stationen und WTs/Produkte ein zeitliches Update der individuellen Planung statt, falls ein Übergang auf einen Bearbeitungsplatz (Übergang 3 in Abbildung 6.3) erfolgt. Dadurch ergibt sich ein Abgleich der geschätzten sowie real eingetretenen Startzeiten. Für das Ende einer Störung aktualisiert die betroffene Station analog ihren individuellen Schedule, um nach der Störung wieder gegenwärtig und mit einer korrekten Planung zu agieren. Die M-Agenten nutzen ihre dezentrale Wissensbasis zur Bewertung von möglichen Auktionsstarts. Die P-Agenten nutzen ihre dezentrale Wissensbasis zur Bewertung der Teilnahme an Auktionen. Die durch Auktionen getroffenen Entscheidungen verteilen die Agenten in bilateraler Kommunikation per MQTT, wodurch wiederum die betroffene Wissensbasis ein Update erfährt. Benötigt die Simulation Entscheidungen zu neuen Zielen für WT/Produkt und FTF (Sockets (h) und (i)), fragt sie diese bei den entsprechenden Node-RED-Instanzen bzw. Agenten der Produktionsentitäten an. Benötigt ein WT einen neuen Produktionsauftrag, findet eine Abfrage über Socket (g) statt und die koordinative VM weist Auftrag sowie FTF zu. Die jeweiligen Antworten der Node-RED-Instanzen werden wiederum über die koordinative VM und die zugeordnete Socket-Verbindung an die Simulation übertragen.

Das auf Steuerungsseite in den unabhängigen Node-RED-Instanzen verwaltete dezentrale Produktionswissen betrifft die **M-Agenten, V-Agenten, P-Agenten** und die **koordinative VM**, welche auch die Auftragseinlastung (**AE-Agent**) koordiniert. Hierzu stellen die in Anhang B.2 dokumentierten Daten die dezentralen Informationen dar, die zur Umsetzung des in Abschnitt 4.3 entwickelten dezentralen Steuerungskonzepts notwendig sind.

Dabei stellen Simulation, Schnittstelle und Steuerung die Aktualität aller Informationen sicher, wodurch die Anforderung an Datenverfügbarkeit erfüllt wird. Einige der Informationen, beispielsweise Stationsfähigkeiten und Transportzeiten, speichern die Node-RED-Instanzen bei der Initialisierung einmalig ab, da sie sich während der Simulation nicht verändern.

6.2.4 Deep Reinforcement Learning-Agent

Der lernende Ansatz nutzt lediglich den Konfigurations-Socket (a) und die RL-spezifische Action-Anfrage über Socket (l), da er für einen performanten Lernvorgang eine schnelle und unkomplizierte Kommunikation benötigt.

Abbildung 6.7 zeigt die Umsetzung des in Kapitel 5 entwickelten lernenden Ansatzes. Mit der in Tecnomatix Plant Simulation implementierten Simulationsumgebung interagiert der PPO-Agent direkt über die Sockets. Der Agent sowie die Umgebung wurden hierzu auf einem Rechner mit einer Intel(R) Core(TM) i7-6820HQ CPU mit 2.7GHz sowie 32 GB Arbeitsspeicher implementiert. Der PPO-Agent nach Alg. 5.1 nutzt die Programmiersprache Python. Die DRL-Umsetzung basiert auf der Machine-Learning-Bibliothek Pytorch [315].

Vor Simulationsstart konfiguriert die Steuerung das Simulationsmodell mittels Socket (a). In der darauffolgenden Simulationsphase löst jedes Simulationsereignis, bei dem ein Fahrzeug

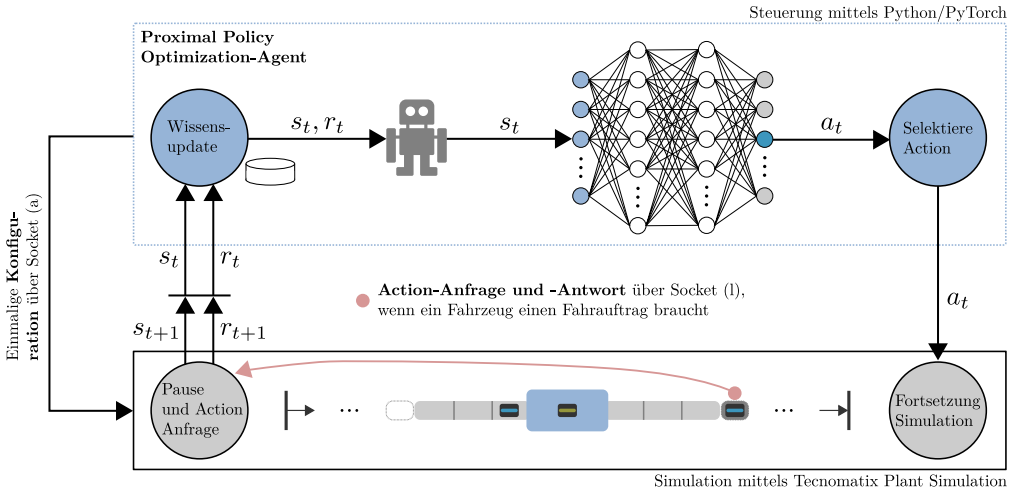


Abbildung 6.7: Umsetzung des lernenden Steuerungsansatzes.

einen neuen Fahrauftrag benötigt, eine Action-Anfrage für das betroffene FTF über Socket (l) aus (vgl. Abbildung 5.6). Dabei sammelt die Tecnomatix Plant Simulation-Umgebung alle für den Agenten notwendigen State-Informationen s_t sowie den Reward r_t (vgl. Abschnitt 5.3.1), stoppt die Simulation und übergibt die Daten an den Agenten. Der Agent speichert s_t und r_t für den Lernprozess. Der State s_t informiert den Agenten über den aktuellen Zustand der Quelle, den Zustand der einzelnen Stationen, den Zustand des anfragenden Fahrzeugs, den Fortschritt des geladenen Produkts (falls Produkt geladen) sowie die validen Actions für die aktuelle Anfrage. Zusätzlich benötigt der Agent sämtliche PPO-Parameter sowie Hintergrundinformationen zur Produktionskonfiguration, zu Transportzeiten, zum Produktmix und zu den zu fertigenden Produkttypen. Auf dieser Datenbasis entscheidet sich der Agent für eine Action a_t und gibt sie über Socket (l) an die Umgebung zurück, woraufhin die Umgebung bis zur nächsten Action-Anfrage weiter simuliert. Diese Zusammenhänge gelten sowohl für die Lern- als auch für die Steuerungsphase. In der Steuerungsphase führt der Agent das Gelernte lediglich aus. NN-Updates finden nicht mehr statt.

6.2.5 Funktionalitätsprüfung mittels Plausibilitätstests

Nach Wenzel et al. [316, S. 169] gelten die Verifikation und Validierung als grundlegendes Qualitätskriterium in Simulationsprojekten im Produktions- und Logistikbereich. Zur Verifikation und Validierung zählt die Prüfung der Plausibilität und damit die Nachvollziehbarkeit der Zusammenhänge sowie die Schlüssigkeit der Ergebnisse [317, S. 22]. Auch Bungartz et al. [318, S. 15] zählen Plausibilitätstests zu gängigen Verfahren zur Prüfung der Korrektheit von Simulationsmodellen. Dieser Abschnitt wendet deshalb Plausibilitätstests an, um sicherzustellen, dass die entwickelten Steuerungsalgorithmen in ihrer implementierten Form korrekte Entscheidungen herbeiführen und ihre Funktionalität gegeben ist. Die Optimalität untersuchen hingegen die späteren Evaluationsabschnitte.

Die implementierten Plausibilitätstests prüfen die Ergebnisse am Ende eines Simulationslaufs. Dabei ist u.a. sicherzustellen, dass die Ergebnisse der VIBN der Steuerung dem Rahmen der gegebenen MP-Konfiguration entsprechen. Hierfür sind folgende Prüfungen durchzuführen:

- **Routenvalidität**: Dieser Test stellt sicher, dass jedes Produkt in der Simulation eine valide Route im Rahmen der gegebenen Prozesspläne und Stationen durchläuft.
- **Prozesszeitenvalidität**: Dieser Test stellt sicher, dass jedes Produkt in der Simulation die der jeweiligen Operation entsprechende Bearbeitungszeit umsetzt.
- **Routenabgleich**: Dieser Test stellt sicher, dass die entschiedene Route eines Produkts auf Steuerungsseite auch der gegangenen Route in der Simulation entspricht.
- **Durchsatz**: Dieser Test stellt sicher, dass sich der erreichte Produktionsdurchsatz in der Simulation in der erwarteten Größenordnung befindet.
- **Schnittstelle**: Dieser Test stellt sicher, dass alle Nachrichten, die Steuerungslogik und Simulation über die Schnittstelle austauschen, ankommen.
- **FTF/WT-Zuweisung**: Dieser Test stellt sicher, dass FTF und WT nach der initialen Zuweisung für die gesamte Simulation gekoppelt bleiben.
- **Produktmix**: Dieser Test stellt sicher, dass der von der Produktionsplanung vorgegebene Produktmix eingehalten wird.

Werden alle Kriterien erfüllt, gelten Simulation, Schnittstelle und Steuerung als plausibel. D.h., sie produzieren korrekte Ergebnisse. Diese Prüfung absolvierte jeder der Steuerungsalgorithmen für jede simulierte MP-Konfiguration erfolgreich.

Derartige Plausibilitätstests können später auch der realen Produktion dienen, die Produktionsabläufe auf Plausibilität zu prüfen, wenn die gesamte Komplexität der Steuerung nicht zu überblicken ist oder die Steuerung eine Blackbox darstellt.

Nach Dokumentation der Implementierung sowie der Funktionalitätsprüfung mittels Plausibilitätstests folgen ansatzspezifische Experimente zur Auslegung der Steuerungen.

6.3 Ansatzspezifische Experimente

Bevor statische, dynamische und Generalisierungsexperimente mit den drei Steuerungsansätzen durchgeführt werden können, sind die Default-Parameter des jeweiligen Ansatzes festzulegen. Dabei wurden einige Parameter bereits in den Algorithmuskapiteln definiert. Die offenen Parametereinstellungen leiten die Abschnitte 6.3.1, 6.3.2 sowie 6.3.3 für den jeweiligen Ansatz ab. Dabei erfolgt die Auslegung unter statischen Bedingungen, da sich

Produktionsszenarien in ihrer Grundstruktur bzw. den daraus resultierenden Eigenschaften wie Transport- und Bearbeitungszeiten nicht maßgeblich unterscheiden. Stationsstörungen werden hingegen in den späteren Experimenten zum Vergleich der Steuerungen untersucht.

6.3.1 Rollierendes Scheduling

Das rollierende Scheduling folgt der Implementierung in Abschnitt 6.2.2. Die einzustellenden Parameter lassen sich in GA-spezifische Parameter sowie Parameter des übergreifenden rollierenden Steuerungsprinzips einordnen.

GA-Parameter

Abschnitt 3.3.2 legt die Codierung, Initialisierung, Decodierung, Evaluation, Selektion, Rekombination und Mutation im verwendeten GA fest. Einzustellen sind außerdem die verbleibenden GA-Stellgrößen aus Tabelle 3.2: die Populationsgröße N_{POP}, die Anzahl der selektierten Individuen N_{SEL}, wobei $N_{\mathrm{POP}} = N_{\mathrm{SEL}}$ gilt, die Anzahl der zu durchlaufenden Generationen N_{GEN} sowie die Wahrscheinlichkeiten für Rekombination und Mutation p_R und p_M. Letztere orientieren sich an der Literatur. p_R beträgt in dieser Arbeit 0.9, p_M beträgt 0.1. Diese Wahrscheinlichkeiten zeigen sich durchweg erfolgreich in [235] und [230], welche ähnliche Scheduling-Probleme lösen. Eine eigene Untersuchung in [29] erzielt damit ebenfalls positive Ergebnisse.

Zur Auslegung der weiteren GA-Parameter berechnet der GA statische Schedules für die MP aus Abschnitt 6.1 im eingeschwungenen Zustand. Hierfür gelten folgende Annahmen:

- **Anzahl Fahrzeuge und WTs**: Im eingeführten Evaluationsbeispiel 1 (Default-MP) sind 21 der 25 Stationen aktiv. Daher wird zur GA-Auslegung die Annahme getroffen, dass sich an jeder dieser 21 Stationen ein Auftrag aus den Produkttypen PT_1 oder PT_2 in Bearbeitung befindet. Dafür benötigt die MP aufgrund der 1:1-Zuweisung von Produkt und WT $n_{\mathrm{WT}} = 21$ WTs sowie $n_V = 21$ Fahrzeuge. Da die Default-MP anhand des berechneten FTF-Minimums mit $n_V = 18$ Fahrzeugen arbeitet, handelt es sich bei 21 Fahrzeugen aufgrund der höheren Komplexität um eine konservative Auslegung.

- **FTF-Geschwindigkeit und Puffer-Parameter**: Hier gelten die Default-Parameter aus Abschnitt 6.1. Ein FTF fährt mit einer Geschwindigkeit von $v_V = 2\ m/s$. Die Puffergrößen für EP und AP sind $n_{\mathrm{EP}} = n_{\mathrm{AP}} = 1$. Die fließbandbedingte Transportzeit pro Pufferplatz ist $t_{\mathrm{PP}} = 3s$.

- **Maximales** ΔT_{GA}: In den späteren Experimenten schaut der GA maximal $300s$ voraus, was geringfügig mehr als drei Mal der Zykluszeit $t_{\mathrm{zyk}} = 95.25s$ entspricht. Je länger die Vorausschau, desto komplexer ist das Problem. Deshalb gilt für eine konservative Auslegung der komplexeste Fall: $\Delta T_{\mathrm{GA}} = 300s$.

Zugrunde liegt also eine MP mit 21 aktiven Aufträgen, die sich an 21 unterschiedlichen Stationen in Bearbeitung befinden und unterschiedlich fortgeschritten in ihrer Abarbeitung sind. Tabelle 6.3 untersucht die Schedule-Berechnung in 10 Parameterkombinationen mit Populationsgrößen zwischen 10 und 100 sowie einer ausreichend groß definierten Anzahl an Generationen von $N_{\text{GEN}} = 100$. In einem 11. Simulationslauf werden die GA-Parameter gezielt höher gestellt ($N_{\text{POP}} = 500$, $N_{\text{GEN}} = 200$), um eine obere Grenze für die Parameterwahl zu definieren. Dabei ist davon auszugehen, dass der 11. Lauf ausreichend gute Ergebnisse zur Orientierung bietet. Jede Parameterkombination berechnet der GA 10 Mal auf einem Rechner mit Intel(R) Core(TM) i7-6820HQ CPU mit 2.7GHz Taktfrequenz und 32 GB Arbeitsspeicher. Die Tabelle zeigt die Durchschnittswerte der jeweils 10 Läufe.

Tabelle 6.3: Parameterkombinationen zur statischen Auslegung der Populationsgröße N_{POP} und der Anzahl zu durchlaufender Generationen N_{GEN} des GAs; durchschnittliche Ergebnisse der jeweils 10 Simulationen.

Nr.	N_{POP}	N_{SEL}	N_{GEN}	p_R	p_M	ΔT_{GA}	$J_{max,m}$	**in Gen. x**
1	10	10	100	0.9	0.1	$300s$	-489.3	78.4
2	20	20	100	0.9	0.1	$300s$	-484.4	78.4
3	30	30	100	0.9	0.1	$300s$	-476.2	77.5
4	40	40	100	0.9	0.1	$300s$	-480.6	60.8
5	50	50	100	0.9	0.1	$300s$	-474.5	76.5
6	60	60	100	0.9	0.1	$300s$	-471.2	62.7
7	70	70	100	0.9	0.1	$300s$	-471.0	61.6
8	80	80	100	0.9	0.1	$300s$	-471.0	60.6
9	90	90	100	0.9	0.1	$300s$	-468.2	71.0
10	100	100	100	0.9	0.1	$300s$	-466.4	69.6
11	500	500	200	0.9	0.1	$300s$	-463.1	41.3

Der erreichte Fitnesswert $J_{max,m}$ der oberen Grenze liegt durchschnittlich bei -463.1. Der GA erreicht diesen bei im Schnitt 41.3 Generationen. Die Experimente zeigen, dass der GA bei höheren Populationsgrößen bessere Ergebnisse erzielt, da er den Suchraum breiter abdeckt, wodurch die Wahrscheinlichkeit guter Individuen steigt. Abbildung 6.8 zeigt einen beispielhaften Verlauf der Optimierung bei $N_{\text{POP}} = 50$. Dabei ist zu sehen, wie sich die durchschnittliche Fitness einer Population sowie die Fitness des jeweils besten Individuums der Generation verbessert und am Ende konvergiert. Die Konvergenz tritt im Bereich zwischen Generation 40 und 60 ein.

Im Vergleich zur oberen Grenze von -463.1 erreicht der GA ab $N_{\text{POP}} = 60$ im Schnitt gute Ergebnisse, deren Fitness weniger als 2% schlechter ausfällt. Allerdings liegen die Rechenzeiten der GA-Berechnungen für Populationsgrößen von mehr als 60 bei mehreren Sekunden und damit nicht im Echtzeitbereich von $T_{max} = 1s$, wenn 100 Generationen zu durchlaufen sind. Hier ist festzuhalten, dass die GA-Implementierung nicht auf Performanz optimiert ist. Es findet beispielsweise keine Parallelisierung der GA-Abläufe statt. Um trotzdem Echtzeitfähigkeit zu erreichen, folgt eine Einführung eines Zeitlimits t_{opt} für die Optimierung.

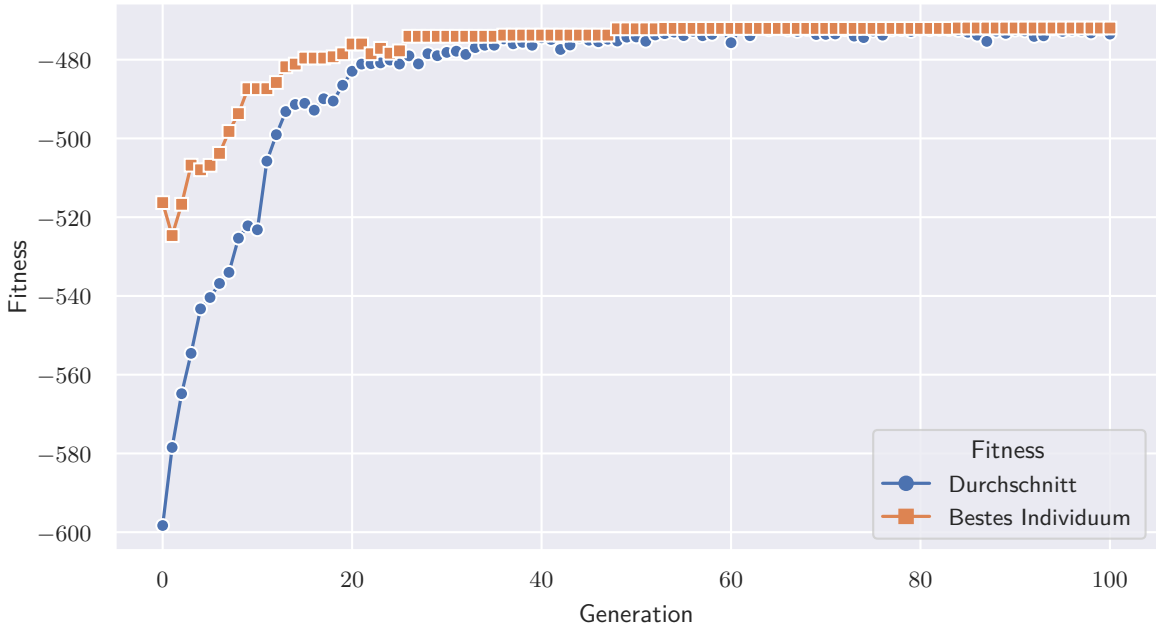


Abbildung 6.8: Verlauf der Fitness für eine GA-Optimierung mit Populationsgröße $N_{\text{POP}} = 50$ bei $N_{\text{GEN}} = 100$ durchlaufenen Generationen.

Tabelle 6.4 wiederholt obige Experimente hierfür und setzt Zeitlimits von $1s$, $2s$ und $3s$. Die Zeitlimits von $2s$ und $3s$ liegen zwar außerhalb der Echtzeitanforderung. Da die Ergebnisse mit $1s$ jedoch nicht ausreichend gut sind, werden sie als zusätzliche Optionen untersucht. Der zentrale Ansatz geht mit den Berechnungszeiten um, indem die berechneten Entscheidungen mit einer Verzögerung von t_{opt} in die Simulation eingespielt werden (vgl. Abschnitt 6.2.2). Jede Simulation erfolgt wiederum 10 Mal. Die Spalte „Gen. x/y“ dokumentiert, in welcher Generation x die Fitness $J_{max,m}$ durchschnittlich erreicht wird und wie viele Generationen y der GA innerhalb des Zeitlimits im Schnitt durchläuft.

Für alle Zeitlimits verbessert sich die erreichte durchschnittliche Fitness $J_{max,m}$ wiederum zuerst bei steigender Populationsgröße. Ab einem gewissen Punkt verschlechtert sie sich jedoch tendenziell wieder. Dieser Grenzpunkt liegt bei $N_{\text{POP}} = 60$. Nur beim $3s$-Zeitlimit existieren bei Populationsgrößen von 80, 90 und 100 noch geringfügig bessere Fitnesswerte. Eine Erhöhung der Populationsgröße über 60 hinaus bietet hierbei keinen wesentlichen Mehrwert, weil bei großen Populationen weniger Generationen innerhalb des Zeitlimits durchlaufen werden. Lediglich beim $3s$-Zeitlimit reichen die rund 18 Generationen noch für bessere Ergebnisse aus. In den wenigen Fällen, in denen bei $t_{\text{opt}} = 1s$ und $t_{\text{opt}} = 2s$ für höhere Populationsgrößen wieder gute Fitnesswerte zu verzeichnen sind, enthalten die ersten GA-Generationen bereits zufallsbasiert gute Individuen. Mit $J_{max,m} = -475.4$ liegt $N_{\text{POP}} = 60$ bei $t_{\text{opt}} = 2s$ zwar um 2.7% unterhalb der oberen Grenze von -463.1, stellt allerdings den besten Kompromiss der Ergebnisse dar, um gute Schedules zu berechnen und sich gleichzeitig

Tabelle 6.4: Parameterkombinationen zur statischen Auslegung der Populationsgröße N_{POP} bei einem Zeitlimit t_{opt} von $1s$, $2s$ und $3s$ für die Optimierung; durchschnittliche Ergebnisse der jeweils 10 Simulationen.

		$t_{\text{opt}} = 1s$		$t_{\text{opt}} = 2s$		$t_{\text{opt}} = 3s$	
Nr.	N_{POP}	$J_{max,m}$	**Gen. x/y**	$J_{max,m}$	**Gen. x/y**	$J_{max,m}$	**Gen. x/y**
1	10	−498.2	41.7/49.9	−501.7	76.3/98.1	−497.7	69.0/97.3
2	20	−496.9	14.3/25.1	−498.0	33.6/48.7	−479.8	50.9/63.0
3	30	−488.5	13.9/17.1	−483.0	24.6/34.2	−481.2	35.3/44.4
4	40	−487.8	9.7/13.6	−482.1	23.3/26.1	−480.9	28.1/35.8
5	50	−487.6	9.0/10.7	−478.6	19.5/21.3	−477.5	24.7/29.4
6	60	−487.3	7.7/9.0	−475.4	17.9/18.6	−477.2	19.5/23.7
7	70	−491.3	6.1/7.5	−480.7	13.5/14.5	−478.7	16.0/18.4
8	80	−490.3	5.7/7.0	−478.0	10.9/12.7	−473.9	16.4/18.2
9	90	−488.3	4.9/6.4	−484.0	9.4/10.3	−476.6	17.9/18.7
10	100	−487.8	4.0/5.5	−478.9	9.1/11.4	−476.8	15.6/18.3
11	500	−489.6	1.3/1.6	−488.1	1.9/2.5	−487.2	2.8/4.0

nicht zu weit von der Echtzeitanforderung zu entfernen. Die Werte für $t_{\text{opt}} = 1s$ sind hierbei beim Bestwert von im Schnitt $J_{max,m} = -487.3$ für $N_{\text{POP}} = 60$ um 2.5% schlechter, wobei der GA durchschnittlich nur neun Generationen durchläuft. Demgegenüber ist der Mehrwert, zum Preis von einer weiteren Optimierungssekunde auf $t_{\text{opt}} = 3s$ zu stellen, anhand der Ergebnisse zu gering.

Deshalb arbeitet der GA in der Folge mit $N_{\text{POP}} = 60$ bei einem Zeitlimit von $t_{\text{opt}} = 2s$ sowie einer maximalen Anzahl von Generationen $N_{\text{GEN}} = 100$ (vgl. Tabelle 6.3). Tabelle 6.4 zeigt hingegen, dass der GA diese Anzahl von Generationen hierbei nicht erreicht. Das Zeitlimit bricht die Optimierung vorher ab und verwendet das bis dato beste Individuum zur Steuerung der MP.

Parameter des rollierenden Schedulings

Der bestimmende Parameter des rollierenden Schedulings ist die Vorausschau ΔT_{GA} und damit die Frage, wie weit der GA für die Entscheidungen in die Zukunft schaut. Damit verbunden ist die Frage, in welchen zeitlich periodischen Abständen Δt_s neue Schedules zu berechnen sind. Außerdem legt ΔT_{AE} für die Auftragseinlastung fest, welche Mindestabstände bei der Einlastung zwischen zwei Einlastungszeitpunkten einzuhalten sind, um Stausituationen zu vermeiden. Für ΔT_{GA} folgt eine ausführliche Herleitung der Auslegung. Δt_s orientiert sich nach (3.4) an der Vorausschau (vgl. Abschnitt 3.3.2). Für ΔT_{AE} hat sich ein empirisch ermittelter Wert von $25s$ als ausreichend erwiesen. Andernfalls würden mit allen initial zur Verfügung stehenden WTs und Fahrzeugen Produkte direkt hintereinander eingelastet werden. Für jede Einlastung erfolgt hierbei eine Schedule-Berechnung. Anschließend warten die eingelasteten Aufträge vor den Stationen der jeweils ersten Operation. Eine Entzerrung dieser Vorgänge mit ΔT_{AE} hat hierbei keinen negativen Einfluss auf das Durchsatzergebnis, da

die eingelasteten Aufträge in jedem Fall vor einer Station warten müssen. In der späteren Einlastung im eingeschwungenen Zustand haben die $25s$ keinen Einfluss, weil die theoretische Zykluszeit der Default-MP bei $95.25s$ liegt, wodurch die Abstände zwischen Einlastungen im Mittel weit über ΔT_{AE} liegen.

Die Zykluszeit der Default-MP ist $95.25s$. Damit der GA mindestens einen Zyklus vorausschaut, wird die untere Grenze der Vorausschau ΔT_{GA} auf $100s$ festgelegt. Die obere Grenze beträgt hingegen $\Delta T_{\mathrm{GA}} = 300s$ als ca. Dreifaches der Zykluszeit. Es gelten die für die MP in Abschnitt 6.1 und den GA in diesem Abschnitt eingeführten Default-Parameter. Abbildung 6.9 dokumentiert für $\Delta T_{\mathrm{GA}} \in \{100s, 150s, 200s, 250s, 300s\}$ die durchschnittlichen Simulationsergebnisse gemittelt über jeweils fünf Simulationsläufe pro Parameterkombination. Dabei produziert die MP 50 Mal PT_1 und 50 Mal PT_2 im 1:1-Produktmix. Stationsstörungen bleiben für die Parameterauslegung aus. Die Abstände zwischen zwei Scheduling-Zeitpunkten berechnen sich in Abhängigkeit der jeweiligen Vorausschau ΔT_{GA} zu $\Delta t_s \in \{50s, 100s, 150s, 200s, 250s\}$. Die Δt_s-Berechnung strebt hierbei unter Berücksichtigung von (3.4) sowie mit einer eingeplanten geringen Sicherheit für jedes ΔT_{GA} so wenige Scheduling-Läufe wie möglich an, um den Einfluss der Rechenzeit zu minimieren. Der Abstand Δt_s stellt lediglich sicher, dass in der Simulation zu jeder Zeit für jedes Produkt ausreichend Entscheidungen zur Verfügung stehen (vgl. Abschnitt 3.3.2). Jedem ΔT_{GA}-Wert ist damit ein fester Δt_s-Wert zugeordnet.

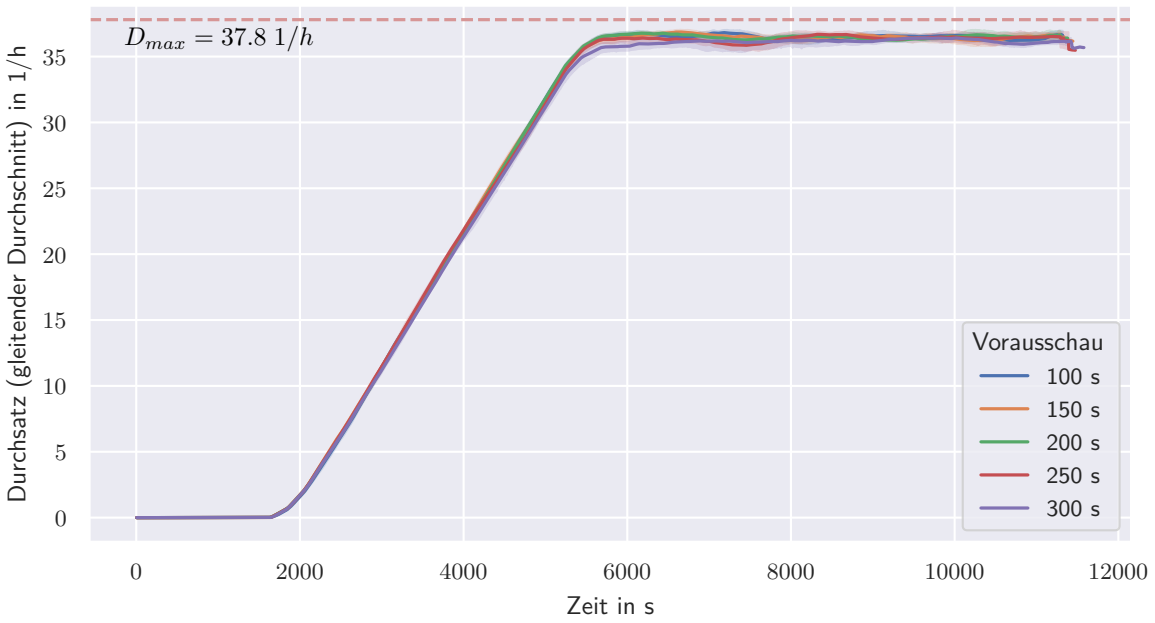


Abbildung 6.9: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP aus Tabelle 6.2 mit 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, für unterschiedliche Vorausschauparameter ΔT_{GA}, fünf Simulationen pro ΔT_{GA}.

Der dargestellte gleitende Durchschnitt des Produktionsdurchsatzes entwickelt sich von 0 in der Anlaufphase bis hin zu Werten zwischen 36.0 $1/h$ und 36.5 $1/h$ im eingeschwungenen Zustand, der nach ca. 6000 Simulationssekunden eintritt. Dabei betrachtet der gleitende Durchschnitt die Durchsatzwerte der jeweils vergangenen 10 Simulationsminuten, was für jegliche Durchsatzkurven im weiteren Verlauf der vorliegenden Arbeit gilt. Die leichte Schattierung hinter den Durchsatzkurven zeigt das 95%-Konfidenzintervall der jeweils fünf Simulationsläufe. Die dargestellten Ergebnisse ähneln sich für die unterschiedlichen Vorausschauzeiten. Der Knick am Ende der Kurven ist auf die Endphase der Simulation zurückzuführen, wobei die letzten Aufträge in Fertigstellung sind, was zu Durchsatzsprüngen führen kann. Bei einer Analyse der Durchsatzdurchschnitte ab $7200s$ schneidet $\Delta T_{\text{GA}} = 200s$ mit $D_m = 36.47\ 1/h$ minimal besser ab als die anderen Vorausschauzeiten, deren Werte mit einer Ausnahme oberhalb von 36.4 $1/h$ liegen. Die Ausnahme ist für $\Delta T_{\text{GA}} = 300s$ zu beobachten. Der Durchsatz reduziert sich auf 36.2 $1/h$, was potenziell an der ansteigenden Problemkomplexität durch eine weitere Vorausschau in Kombination mit dem eingestellten Zeitlimit liegt. Obwohl für ein größeres ΔT_{GA} ein höherer Durchsatz zu erwarten wäre, weil der GA gesamtheitlicher optimiert, führt das Zeitlimit bei höherer Problemkomplexität zu weniger durchlaufenen GA-Generationen. Analog zur Untersuchung in Tabelle 6.4 bietet sich deshalb auch hier eine Kompromissbildung für die Parameter an, weshalb eine Entscheidung für $\Delta T_{\text{GA}} = 200s$ bei zugeordnetem $\Delta t_s = 150s$ fällt. $\Delta T_{\text{GA}} = 200s$ erzielt zum einen die besten Ergebnisse und liegt zum anderen in der Mitte der untersuchten Bandbreite, wodurch ein Kompromiss aus gesamtheitlicher Optimierung und Anzahl durchlaufener GA-Generationen möglich ist. Gleichzeitig ist jedoch festzuhalten, dass der erreichte Durchsatz von $D_m = 36.47\ 1/h$ 3.5% unterhalb des theoretischen Optimums $D_{max} = 37.8\ 1/h$ liegt.

Schließlich stellt sich die Frage, wie viel der zentrale Ansatz durch die Einführung des Zeitlimits von $t_{\text{opt}} = 2s$ an Lösungsgüte verliert. Zum Vergleich erreichen Simulationen, in denen die GA-Rechenzeit ignoriert wird und nicht in die Simulation einfließt, Durchsatzwerte von bis zu 36.63 $1/h$ für dieselbe Konfiguration. Dabei durchläuft der GA bei jeder Berechnung alle $N_{\text{GEN}} = 100$ Generationen. Statt der Rechenzeit repräsentiert damit N_{GEN} das Abbruchkriterium. Die zugehörigen Ergebnisse dokumentiert Anhang C.1. Somit führt die Einführung des Zeitlimits von $t_{\text{opt}} = 2s$ lediglich zu Einbußen von ca. 0.4% im Vergleich zum idealisierten Szenario. Die durchgeführte GA-Auslegung wird deshalb als legitim angenommen.

Die Default-Parameter des zentralen Ansatzes ergeben sich ergänzend zu Abschnitt 3.3 zu Tabelle 6.5.

Für jede simulierte Parameterkombination in den folgenden Evaluationsabschnitten werden fünf Simulationen durchgeführt, da sich die Simulationsläufe aufgrund der Stochastizität des GAs und der realen Kommunikation zwischen den VMs unterscheiden. Bei jeder Simulation verwendet der GA dabei eine zufällige Seed-Einstellung. Mehr als fünf Läufe bieten hierbei keinen wesentlichen Erkenntnisgewinn, was Anhang C.2 zeigt.

Tabelle 6.5: Default-Parameter des zentralen Steuerungsansatzes.

Stellgröße	**Default-Wert**
Populationsgröße N_{POP}	60
Anzahl selektierter Individuen N_{SEL}	60
Anzahl Generationen N_{GEN}	100
Zeitlimit für Optimierung t_{opt}	$2s$
Rekombinationswahrscheinlichkeit p_R	0.9
Mutationswahrscheinlichkeit p_M	0.1
Vorausschau des GAs ΔT_{GA}	$200s$
Abstand Schedule-Berechnung Δt_s	$150s$
Mindestabstand Einlastungen ΔT_{AE}	$25s$

6.3.2 Auktionsbasiertes Multiagentensystem

Das auktionsbasierte MAS folgt der Implementierung in Abschnitt 6.2.3. Der einzustellende Parameter beim auktionsbasierten MAS reduziert sich zur Vorausschau ΔT_{MAS} für das Anbieten freier Kapazität pro aktiver Station der MP. Für die Default-MP aus Abschnitt 6.1 untersucht Abbildung 6.10 in Analogie zum zentralen Ansatz verschiedene Werte für ΔT_{MAS}. Dabei produziert die MP 50 Mal PT_1 und 50 Mal PT_2 im 1:1-Produktmix. Wiederum fallen keine Stationsstörungen in der Auslegung an. ΔT_{MAS} nimmt Werte zwischen $0s$ (reines Pullprinzip) und $100s$ (ca. einmal Zykluszeit) in $10s$-Schritten an. Wie die Ergebnisse zeigen, liefert eine noch weitere Vorausschau keinen Mehrwert. Für jedes ΔT_{MAS} erfolgen fünf Simulationsläufe. Die Abbildung zeigt die durchschnittlichen Verläufe des Durchsatzes D_m.

Für $\Delta T_{\text{MAS}} = 0s$ startet eine Station erst eine neue Auktion, wenn der aktuelle Auftrag vollständig abgearbeitet ist und damit keine weiteren Operationen geplant sind (Pullprinzip, wenn Station leer). Dadurch steigen die Durchlaufzeiten der Aufträge, weil das die Auktion gewinnende Produkt jeweils erst zur auktionierenden Station zu transportieren ist, bevor die Bearbeitung beginnen kann. Die Durchlaufzeiterhöhung führt zu einer Verringerung des erreichten Durchsatzes. Für $\Delta T_{\text{MAS}} = 10s$ kann die Transportzeit, die nach einem Auktionsgewinn anfällt, schon teilweise kompensiert werden. Der erreichte Durchsatz steigt im eingeschwungenen Zustand durchschnittlich von $D_m = 27.78\ 1/h$ auf $D_m = 32.80\ 1/h$. Für $\Delta T_{\text{MAS}} = 20s$ findet ein weiterer Durchsatzsprung auf $D_m = 36.10\ 1/h$ statt. Das Maximum erreicht der dezentrale Ansatz bei $\Delta T_{\text{MAS}} = 30s$ mit $D_m = 36.45\ 1/h$. Die weiteren Verläufe ähneln sich sehr stark. Für höhere Werte nimmt der Durchsatz wieder ab. Vor allem bei $\Delta T_{\text{MAS}} = 90s$ sowie $\Delta T_{\text{MAS}} = 100s$ sinkt der Durchsatz auf $36.07\ 1/h$ und $35.93\ 1/h$. Die zugehörigen Durchsatzergebnisse dokumentiert Anhang C.3.

Hierbei gilt: Je größer die Vorausschau, desto mehr Verwirbelungen entstehen in den Pro-

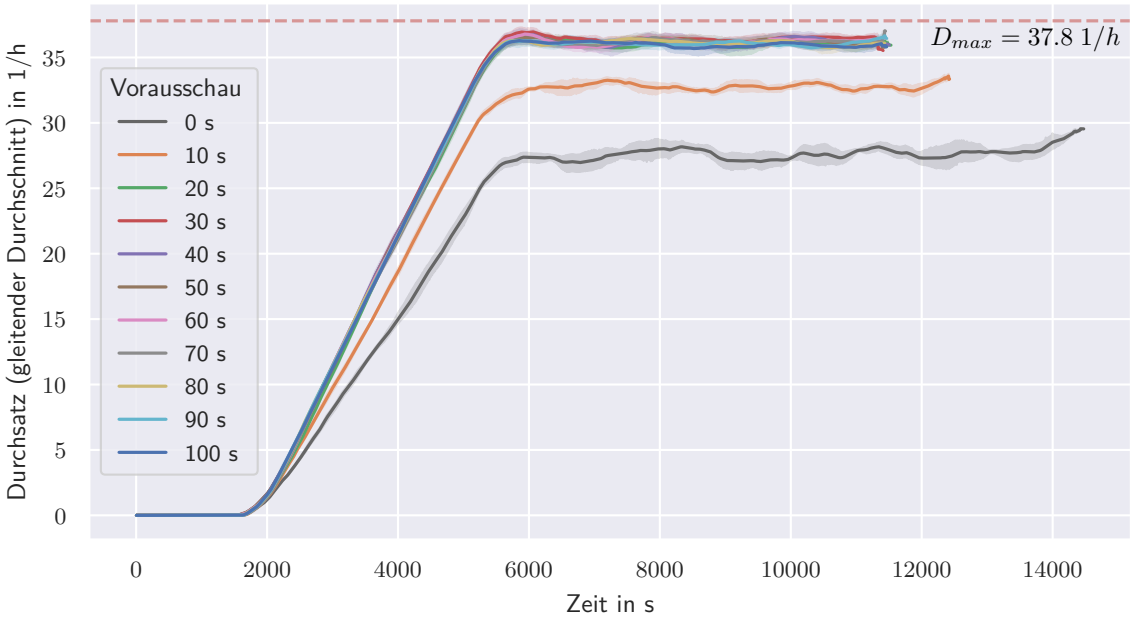


Abbildung 6.10: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP aus Tabelle 6.2 mit 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, für unterschiedliche Vorausschauparameter ΔT_{MAS}, fünf Simulationen pro ΔT_{MAS}.

duktionsabläufen. Wenn eine Station $100s$ vorausblickt, nehmen an der Auktion lediglich die Produkte teil, welche sich bereits an einer Station in Bearbeitung befinden. Das höchste Gebot gewinnt dabei die Auktion. Die Entscheidung wird nur im Störungsfall des gewinnenden Produkts revidiert. I.d.R. steht sie also fest. Ein Produkt, welches erst später zur Teilnahme an der Auktion infrage kommt, jedoch beispielsweise früher an der Station ankäme, hat keine Möglichkeit ihren Wert für die Auktion zu zeigen. Bei kleineren Vorausschauzeiten können damit mehr Produkte an der Auktion teilnehmen. Dadurch findet die erwünschte Koordination zwischen den Produkten statt. Die zu lösenden lokalen Probleme erreichen einen etwas globaleren Blick auf das Gesamtproblem.

Schlussfolgernd ist bei Einstellung von ΔT_{MAS} ein Kompromiss zu finden. Die Vorausschau muss groß genug sein, damit anfallende Transportzeiten kompensiert werden können. Gleichzeitig ist die Vorausschau klein genug zu wählen, um die Koordination zwischen konkurrierenden Produkten sicherzustellen. Dieser Kompromiss stellt sich bei $\Delta T_{\text{MAS}} = 30s$ mit $D_m = 36.45\ 1/h$ ein. Deshalb fällt die Entscheidung auf $\Delta T_{\text{MAS}} = 30s$ als Default-Parameter für den dezentralen Steuerungsansatz, dokumentiert in Tabelle 6.6.

Tabelle 6.6: Default-Parameter des dezentralen Steuerungsansatzes.

Stellgröße	**Default-Wert**
Vorausschau pro Station ΔT_{MAS}	$30s$

Für jede simulierte Parameterkombination in den folgenden Evaluationsabschnitten werden fünf Simulationen durchgeführt, da sich die Simulationsläufe aufgrund der realen Kommunikation zwischen den VMs unterscheiden. Mehr als fünf Läufe bieten hierbei keinen wesentlichen Erkenntnisgewinn, was Anhang C.4 zeigt.

6.3.3 Deep Reinforcement Learning-Agent

Der lernende Ansatz folgt der Implementierung in Abschnitt 6.2.4. Zur Umsetzung des lernenden PPO-Algorithmus sind zahlreiche Hyperparameter einzustellen. Da eine Analyse des gesamten Parameterraums, beispielsweise als Rastersuche, nicht durchführbar ist, wie auch die analysierten Ansätze in Abschnitt 5.1.2 zeigen, orientiert sich die Arbeit an der PPO-Literatur sowie an empirischen Vorversuchen. Tabelle 6.7 fasst dabei die resultierenden Default-Parameter des lernenden Ansatzes zusammen, welche durch vorsichtiges Optimieren bis zu einer stabilen Zielerreichung des Agenten erarbeitet wurden.

Tabelle 6.7: Default-Parameter des lernenden Steuerungsansatzes.

Stellgröße	**Default-Wert**
Lernrate α	$2 \cdot 10^{-3}$
Clip-Parameter ϵ	0.2
Value-Funktions-Koeffizient c_1	0.5
Entropie-Koeffizient c_2	0.01
GAE-Parameter λ	0.95
Discount-Faktor γ	0.9
Trajektorienlänge n_τ	1000
Episodenanzahl n_{episodes}	$1000, 2000$
NN-Updates pro Trajektorie n_{epochs}	3
Anzahl Umgebungen n_{env}	$2, 4, 6$
Bestrafungs-Reward r_{punish}	-20
Bonus-Reward r_{bonus}	+1
Optimierer	Adam
NN-Konfiguration	123–160-80–29
Geteiltes NN	Ja
Action Masking	Ja

Die folgenden Absätze vertiefen die Auswahl der NN-Konfiguration und die Verwendung von Action Masking zur Steigerung der Performanz des Algorithmus. Darauffolgend geht eine Untersuchung auf die Anzahl der Umgebungen n_{env} sowie die Anzahl der gesammelten Episoden n_{episodes} in der Lernphase des Agenten ein.

NN-Konfiguration

Bei der NN-Konfiguration sind die Input- und die Output-Schicht durch das State- und Action-Design aus Abschnitt 5.3.1 in Abhängigkeit der zu steuernden Produktion definiert.

Für das Beispiel aus Abschnitt 6.1 gilt: $h_{\text{in}} = 123$ Input-Neuronen repräsentieren die Anzahl der State-Informationen. $h_{\text{out}} = 29$ Output-Neuronen repräsentieren 28 mögliche Actions des Actors sowie die Approximation der Value-Funktion des Critics mithilfe eines Neurons. Die Auslegung der versteckten Schichten basiert auf Schröder [292, S. 81f.] und Heaton [319, S. 127ff.], wobei sich beide Autoren einig sind, dass die Netzauslegung problemspezifisch ist und durch Experimentieren anzunähern ist. Ein NN mit zwei versteckten Schichten ermöglicht die Approximation beliebiger Funktionen. Gleichzeitig gibt es laut Heaton [319, S. 128] keine Notwendigkeit, mehr als zwei versteckte Schichten zu verwenden. Zu wenige Schichten bzw. Neuronen pro Schicht verringern hingegen die Repräsentationsfähigkeit, während zu viele Schichten bzw. Neuronen die Verallgemeinerungsfähigkeit reduzieren [292, S. 82]. Diesen Richtlinien folgend, arbeitet die NN-Konfiguration mit zwei versteckten Schichten. Durch empirische Annäherung von größeren Schichten kommend, ergibt sich die Größe der ersten versteckten Schicht zu 160 sowie die der zweiten Schicht zu 80 Neuronen.

Action Masking

Abbildung 6.11 verdeutlicht die Performanzsteigerung durch die Implementierung von Action Masking in der Lernphase des PPO-Agenten. Dabei lernt der Agent die Steuerung der Default-MP aus Abschnitt 6.1, ohne dass dabei Stationsstörungen auftreten. Der Graph zeigt die Entwicklung des gesammelten Rewards pro Episode für insgesamt $n_{\text{episodes}} = 1000$ Episoden, wobei der Agent mit $n_{\text{env}} = 2$ parallelen Umgebungen interagiert. Ohne Action Masking sind Bestrafungs-Rewards r_{punish} aktiv, wenn der Agent nicht valide Actions wählt (vgl. (5.46)). Deshalb startet die Kurve stärker negativ. Am Ende lernt der Agent, keine validen Actions zu wählen, kommt dem Optimum näher, bleibt allerdings deutlich unterhalb der mit Action Masking erreichten Rewards. Aufgrund der Verzerrung beim Lernvorgang durch die Bestrafungs-Rewards scheint der Agent dabei ein lokales Optimum zu lernen. Die Action Masking-Kurve konvergiert auf einem wesentlich höheren Reward-Wert, weil der Agent sich auf das Wesentliche konzentrieren kann und die Verzerrung durch Bestrafungs-Rewards entfällt. Am Ende pendeln sich die in einer Episode gesammelten Rewards bei knapp weniger als -200 ein. Damit hat der Agent Zeitverschwendung in der MP minimiert, wodurch sich der Durchsatz maximiert. Hierbei erzielt der Ansatz mit Action Masking einen Durchsatz von $D_m = 36.28\ 1/h$. Ohne Action Masking hat die Simulation hingegen keinerlei Ausbringung. Dieses Ergebnis ist auf die Anlaufphase der Produktion zurückzuführen. Für die Anlaufphase sammelt der Agent hierbei nur wenige Erfahrungen, weil sie ausschließlich direkt zu Beginn des Lernprozesses auftritt. Zu diesem Zeitpunkt wählt der Agent noch einige invalide Actions und wird dafür mit r_{punish} bestraft. Aufgrund der wenigen Beispielerfahrungen reichen diese Bestrafungs-Rewards allerdings nicht aus, dem Agenten für die Anlaufphase valide Actions beizubringen. Außerdem fehlen valide Beispielerfahrungen. Deshalb wählt er in der Evaluationssituation, wenn die Produktion neu gestartet wird, ausschließlich nicht

valide Actions und die Ausbringung ist $D_m = 0$.[3] Der deutliche Performanzunterschied bekräftigt die Implementierung von Action Masking. Diese Erkenntnis bestätigt die Ergebnisse in [302], [303] hinsichtlich der Performanzverbesserung durch Action Masking. Ergänzend dokumentiert Anhang C.5 die Entwicklung der gesammelten Rewards sowie die Entwicklung des Leistungsmaßes über die Lernzeit bei aktivem Action Masking.

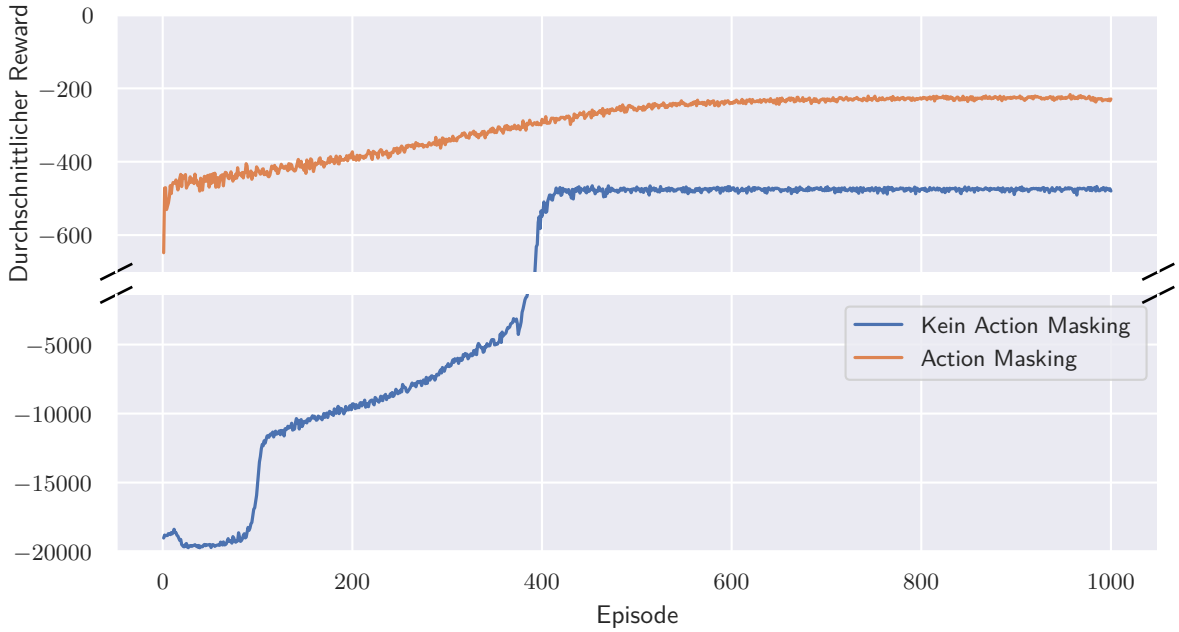


Abbildung 6.11: Lernphase der Steuerung der Default-MP aus Tabelle 6.2 mit und ohne Action Masking.

Anzahl paralleler Umgebungen

In der Lernphase interagiert der Agent mit n_{env} parallelen Umgebungen. In Abhängigkeit von n_{env} untersucht Tabelle 6.8 die erzielten Durchsatzergebnisse in der eingeschwungenen Simulation für die Steuerung der Default-MP ohne Störungen für zwei, vier und sechs Umgebungen. Dabei repräsentieren die simulierten Durchsatzwerte den Durchschnitt auf Basis von jeweils fünf Lernvorgängen, welche durch unterschiedliche, vordefinierte Seed-Einstellungen auf Agentenseite abgebildet werden. Jede Lernphase dauert $n_{\text{episodes}} = 1000$ Episoden. Die Konfiguration mit zwei Umgebungen schneidet mit $D_m = 36.79\ 1/h$ geringfügig besser ab als die anderen beiden Konfigurationen, wobei die Unterschiede geringer als 0.35% sind. Obwohl eine geringere Umgebungszahl weniger Lernerfahrungen bereitstellt, erzielen hier zwei Umgebungen die besten Ergebnisse. Ein zusätzlicher Vorteil von lediglich zwei Umgebungen ist der schnellere Lernvorgang, weil weniger Interaktion notwendig ist. Auch 1000 Episoden

[3] Die Konvergenzkurve in Abbildung 6.11 führt ohne Action Masking trotzdem zum Lernen nicht valider Actions, allerdings im eingeschwungenen Zustand (Continuing-Task), der die spätere Lernphase betrifft. Dies lässt allerdings die Anlaufphase in der Evaluationssituation nicht überwinden.

scheinen eine vorerst ausreichende Lerndauer für den Produktionsfall ohne Störungen unter den definierten Default-Parametern des Agenten darzustellen, was die Lernkurven in Anhang C.6 verdeutlichen. Mit $n_\tau = 1000$ Actions pro Trajektorie sammelt der Agent bei zwei Umgebungen und 1000 Episoden insgesamt $n_{\text{env}} \cdot n_{\text{episodes}} \cdot n_\tau = 2 \cdot 10^6$ Actions zum Lernen der Policy. Vorerst gelten $n_{\text{env}} = 2$ und $n_{\text{episodes}} = 1000$ als Default-Parameter. Die Anzahl der verwendeten Umgebungen n_{env} sowie die Anzahl der gesammelten Episoden n_{episodes} unterliegen jedoch Variationen in den späteren Experimenten, wenn sich die Problemkomplexität durch häufige Störungen und eine veränderte Produktionskonfiguration ändert. Für diese Fälle sind für ein optimales Lernverhalten mehr Lerndaten erforderlich. Anhang C.7 dokumentiert die Durchsatzverläufe der Simulationen aus Tabelle 6.8.

Tabelle 6.8: Durchschnittliche Durchsätze D_m im eingeschwungenen Zustand für die Simulationszeit ab $7200s$ in Abhängigkeit der Anzahl der Umgebungen n_{env}, fünf Lern-Seeds pro n_{env}.

Anzahl Umgebungen n_{env}	**Durchsatzdurchschnitt** D_m
2	36.79 $1/h$
4	36.66 $1/h$
6	36.69 $1/h$

Für jede simulierte Parameterkombination in den folgenden Evaluationsabschnitten lernt der Agent mit fünf unterschiedlichen Seed-Einstellungen, was zu fünf verschiedenen, an die jeweilige Lernphase anschließenden Simulationsläufen führt. Im jeweiligen Simulationslauf führt der PPO-Agent das Gelernte aus. Mehr als fünf Seed-Einstellungen bieten hierbei keinen wesentlichen Erkenntnisgewinn, was Anhang C.8 zeigt.

6.4 Vergleichende Experimente im Evaluationsbeispiel

Für die vergleichenden Experimente nutzen die drei Algorithmen zunächst die jeweils abgeleitete Default-Konfiguration aus Abschnitt 6.3. Zum Vergleich der Simulationsergebnisse dienen das in Abschnitt 6.1 berechnete theoretische Durchsatzoptimum von $D_{max} = 37.8\ 1/h$ sowie folgende Heuristik, die der SQ-Heuristik ähnelt und sich bei jedem Produktionsschritt für den kürzesten Weg entscheidet:

$$\min_{M_k \in M_{ipl}} \Big(\max(t_{\text{TR},Pos \to M_k} + t_{\text{EP},min},\ t_{ipl,k,\text{Rest}} + \sum_{i=0}^{n_{J,M_k}} t_{ipl,k}) \Big). \tag{6.3}$$

Die Heuristik evaluiert damit die zur Verfügung stehenden Alternativstationen M_{ipl} für eine Operation O_{ipl}, indem sie für jede Station den frühestmöglichen Zeitpunkt berechnet, zu dem das Produkt in Bearbeitung gehen könnte. Hierzu vergleicht sie die Transportzeit $t_{\text{TR},Pos \to M_k}$ zu Station $M_k \in M_{ipl}$, einschließlich der fließbandbedingten Mindest-Transportzeit im Puffer $t_{\text{EP},min}$, mit der verbleibenden Bearbeitungszeit aller an Station M_k

eingeplanten Prozesse. Dabei umfasst n_{J,M_k} die im EP oder vor der Station M_k wartenden bzw. sich auf dem Weg zur Station befindlichen Aufträge. Die Heuristik entscheidet sich für die Stationsalternative mit der frühestmöglichen Bearbeitung. Die Auftragseinlastung folgt dem von der Produktionsplanung vorgeschlagenen Produktionsprogramm, wobei die Aufträge bei Verfügbarkeit eines FTFs und eines WTs direkt nacheinander eingelastet werden.

Abschnitt 6.4.1 vergleicht die Algorithmen unter statischen Bedingungen. Abschnitt 6.4.2 befasst sich anschließend mit dynamischen Produktionsbedingungen und berücksichtigt Stationsstörungen. Deshalb wird eine Simulation ohne Störungen in der Folge als statisch bezeichnet, wohingegen Simulationen mit Störungen dynamisch sind.

6.4.1 Optimalität ohne Störungen

Zur Analyse der Optimalität der Algorithmen unter statischen Bedingungen variiert dieser Abschnitt die Anzahl der Fahrzeuge und damit WTs ($n_V = n_{\mathrm{WT}}$) sowie die FTF-Geschwindigkeit v_V im Vergleich zur Default-MP aus Abschnitt 6.1 (Evaluationsbeispiel 1). Eine Variation der Puffergrößen ist hingegen nicht relevant, weil Fahrzeuge auch vor der Station warten und damit als Zusatzpuffer agieren können (vgl. Abschnitt 2.4.2). Die Pufferkapazitäten von $n_{\mathrm{EP}} = n_{\mathrm{AP}} = 1$ bleiben für alle weiteren Simulationen konstant. Die Pufferung eines WTs mit jeweils einem Pufferplatz ist hierbei notwendig, um die Übergabe des WTs an die Station sowie die Aufnahme des WTs nach der Bearbeitung von der Bearbeitung selbst zu entkoppeln.

Die Variation der Fahrzeug/WT-Zahl sowie der FTF-Geschwindigkeiten ermöglicht einen Einblick in die Sensitivität der Kombination aus Steuerung und MP, bezogen auf den maximalen Produktionsbestand ($PB_{max} = n_V = n_{\mathrm{WT}}$) sowie auf den Einfluss der Transportzeiten (v_V). Außerdem lässt sich feststellen, ob und mit welcher MP-Konfiguration eine Steuerung das theoretische Optimum erreicht.

Tabelle 6.9 fasst die Simulationsergebnisse zusammen. Parameterkombination fünf stellt die Default-MP mit 18 Fahrzeugen und einer Geschwindigkeit von 2 m/s dar. Für $n_V \in \{17, 18, 19\}$ erfolgen jeweils Simulationen zu $v_V \in \{1\ m/s, 2\ m/s, 3\ m/s\}$. Die Geschwindigkeiten orientieren sich hierbei an der Realität. In Produktionssystemen, in denen sich auf der Produktionsfläche auch Personen bewegen, sind aus Sicherheitsgründen niedrigere Werte einzuhalten. Sind weniger Menschen unterwegs und der notwendige Platz vorhanden, können höhere Geschwindigkeiten die Transportzeiten reduzieren. Die dokumentierten Durchsatzwerte D_m repräsentieren den Durchschnitt von jeweils fünf Simulationsläufen im eingeschwungenen Zustand der MP ab einer Simulationszeit von $7200s$. Die Varianz der zentralen und dezentralen Steuerung ergibt sich aus der realen Kommunikation zwischen den VMs, wobei im zentralen Ansatz zusätzlich die Stochastizität des GAs wirkt. Die Varianz im lernenden Ansatz wird durch fünf unterschiedliche Seeds auf Agentenseite erzielt. Die blaue Markierung hebt das beste Ergebnis der jeweiligen Parameterkombination hervor.

Insgesamt lassen sich wie erwartet höhere Durchsatzwerte feststellen, wenn mehr Fahrzeuge bzw. höhere Geschwindigkeiten in Betrieb sind. Höhere Geschwindigkeiten verringern mit den Transportzeiten die Durchlaufzeiten und steigern damit indirekt den Durchsatz. Mehr Fahrzeuge erhöhen den Produktionsbestand, wodurch nach Abbildung 2.4 ebenfalls der Durchsatz steigt.

Tabelle 6.9: Statische Simulationsergebnisse der drei Algorithmen im Vergleich zur eingeführten Heuristik in Abhängigkeit der Anzahl der Fahrzeuge n_V und der FTF-Geschwindigkeit v_V, fünf Simulationen pro Parameterkombination.

			Zentral	**Dezentral**	**Lernend**	**Heuristik**
Nr.	n_V	v_V $[m/s]$	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$
1	17	1	31.58	30.63	32.05	32.08
2	17	2	35.23	35.21	35.49	35.92
3	17	3	36.10	36.26	36.48	36.54
4	18	1	33.11	32.44	33.69	33.77
5	18	2	36.47	36.45	36.79	36.93
6	18	3	37.46	37.70	37.46	37.60
7	19	1	34.58	33.27	35.27	35.46
8	19	2	37.46	37.80	37.80	37.80
9	19	3	37.79	37.80	37.80	37.80

Abschnitt 6.1 leitet nach dem Gesetz von Little ab, dass 18 Fahrzeuge mit einer Geschwindigkeit von 2 m/s zur Erreichung des theoretischen Durchsatzoptimums von 37.8 $1/h$ für die gegebene Default-MP unter idealen Bedingungen ausreichen. Für diese Konfiguration erreicht keiner der Algorithmen das theoretische Optimum, wie Versuchsreihe fünf zeigt. Am besten agiert die Heuristik mit 36.93 $1/h$, jedoch um 2.3% schlechter als das Optimum.

Insgesamt zeigt sich die Heuristik in nahezu allen Konfigurationen als beste Lösung. Dies deutet auf eine hohe Güte der Heuristik in der gegebenen Produktionsumgebung hin. Da insgesamt Durchlaufzeiten minimiert werden, um den Durchsatz zu maximieren, erzielt die Heuristik hierbei gute Ergebnisse, indem sie pro Produktionsschritt den kürzeren Weg wählt. Die Problemkomplexität hält sich unter statischen Bedingungen in der Default-MP jedoch in Grenzen, da pro Produktionsschritt der Produkte meist zwei Optionen zur Auswahl stehen. Später verändert sich dieses Verhältnis durch das Einbringen von Stationsstörungen.

Das Güte-Verhältnis zwischen dem zentralen, dezentralen und lernenden Ansatz variiert pro Parameterkombination. Der zentrale Ansatz erreicht jedoch durchgängig einen geringfügig niedrigeren Durchsatz als die lernende Steuerung. Der dezentrale Ansatz agiert in einem Fall sogar besser als die Heuristik, wobei tendenziell der lernende Ansatz leicht bessere Ergebnisse als das MAS liefert. Der dezentrale Ansatz fällt außerdem deutlicher ab, wenn die FTF-Geschwindigkeit 1 m/s beträgt. Aufgrund der daraus resultierenden längeren Transportzeiten reicht die Vorausschau von $\Delta T_{\text{MAS}} = 30s$ für diese Fälle nicht aus, um die anfallenden Transportzeiten vollständig zu kompensieren. Da die weiteren Experimente eine

Geschwindigkeit von 1 m/s nicht weiter untersuchen, erfolgt hier keine tiefergehende Analyse mit größerer Vorausschau.

Das theoretische Durchsatzoptimum erreichen die Heuristik sowie der dezentrale und lernende Ansatz zuverlässig bei 19 Fahrzeugen mit 2 m/s, was für die Lösungsgüte aller drei Ansätze spricht. Dabei brauchen die Algorithmen also ein Fahrzeug mehr, als theoretisch berechnet. Der zentrale Ansatz erreicht das Optimum allerdings erst, wenn die Fahrzeuggeschwindigkeit auf 3 m/s erhöht wird. Da dies eine Ausnahme zu den anderen Algorithmen darstellt, erhöht sich die Anzahl an Fahrzeugen und WTs in der Folge auf $n_V = n_{\mathrm{WT}} = 19$. Die Fahrzeuggeschwindigkeit bleibt bei $v_V = 2\ m/s$. Die zugehörigen Durchsatzverläufe über die Simulationszeit sind in Anhang D.1 dokumentiert.

Für die Konfiguration aus 19 Fahrzeugen mit Geschwindigkeiten von 2 m/s zeigt Abbildung 6.12 die Verteilung der Durchlaufzeiten der fünf Simulationen pro Ansatz für die beiden Produkttypen als Boxplot. Die schwarz eingerahmten weißen Kreise markieren den jeweiligen Durchschnittswert. Dabei fällt auf, dass der dezentrale Ansatz wesentlich niedrigere Durchlaufzeiten erreicht: im Mittel für beide Produkttypen 1691.44s. Der zentrale und lernende Ansatz sowie die Heuristik liegen mit 1875.19s, 1892.28s und 1870.51s nahe zusammen.

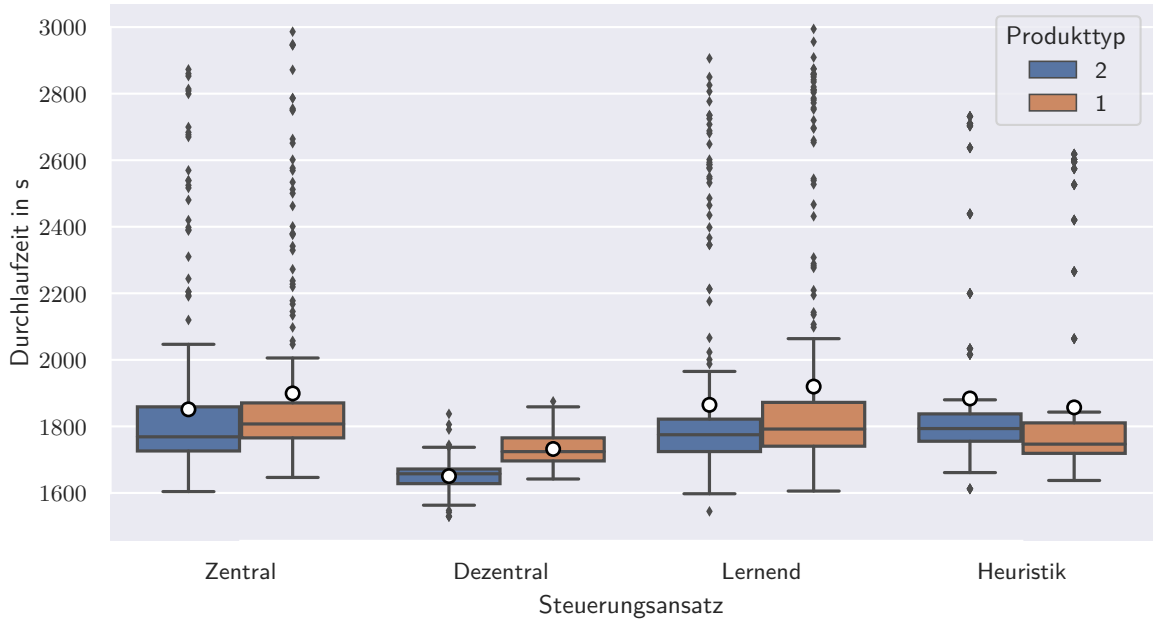


Abbildung 6.12: Durchlaufzeitverteilungen aller Steuerungsansätze über alle fünf Simulationsläufe pro Ansatz für die Simulation der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$ und $v_V = 2\ m/s$.

Die auftragsbezogenen Durchlaufzeiten des dezentralen Ansatzes sind hierbei geringer, weil sich der Zeitpunkt der Auftragseinlastung eines Auftrags erst durch den Gewinn einer Auktion entscheidet, während die anderen Ansätze die Aufträge direkt starten, wenn dies möglich ist. Letzteres führt zu längeren Wartezeiten vor der ersten Bearbeitung, was zwar die

Durchlaufzeit verlängert, jedoch nicht den Durchsatz beeinträchtigt. Diese Stausituationen lösen sich dann durch die Einschwingphase der Produktion auf. Trotzdem zeigen sie sich in der Durchlaufzeitverteilung der zentralen, lernenden und heuristischen Steuerung. Die Ausreißer nach oben repräsentieren hierbei die jeweilige Durchlaufzeitverlängerung durch die initialen Wartezeiten. Aus diesen Gründen erzielt der dezentrale Ansatz, obwohl er niedrigere Durchlaufzeiten erreicht, keine höheren Durchsätze im Vergleich zu den anderen Steuerungen. Bevorzugt ein Anwender trotzdem niedrige Durchlaufzeiten und niedrige Bestände, bietet der dezentrale Ansatz in der aktuellen Ausführung Vorteile.

Diese Vorteile zeigt auch Abbildung 6.13. Sie stellt die Entwicklung der Produktionsbestände für einen der fünf Simulationsläufe pro Algorithmus dar. Dabei ist der maximale Produktionsbestand $PB_{max} = n_V = 19$ von der Anzahl der Fahrzeuge begrenzt. Dieser maximale Bestand baut sich für die zentrale, lernende und heuristische Steuerung schneller auf, wobei der zentrale Ansatz durch die Einhaltung des Mindestabstands ΔT_{AE} bei der Einlastung etwas langsamer agiert. Die orange-farbene Kurve der dezentralen Steuerung zeigt den langsamsten Bestandsaufbau durch oben erklärten Auktionsmechanismus.

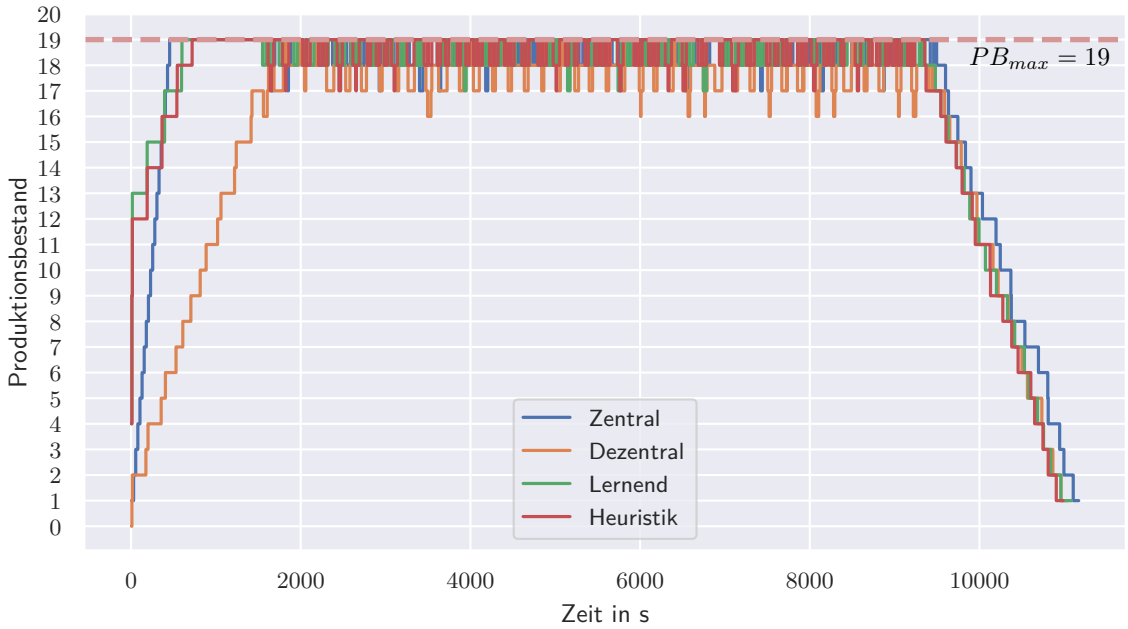


Abbildung 6.13: Entwicklung des Produktionsbestands für alle Steuerungsansätze über jeweils einen der fünf Simulationsläufe der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$ und $v_V = 2\ m/s$.

Zusammenfassend lässt sich feststellen, dass die eingeführte Heuristik für einfache statische Umgebungen gute Ergebnisse generiert und sogar meist besser agiert als die komplexeren Steuerungsansätze. Bei 19 Fahrzeugen mit 2 m/s erreichen die dezentrale, lernende und heuristische Steuerung das theoretische Durchsatzoptimum. Der zentrale Ansatz braucht hierfür eine FTF-Geschwindigkeit von 3 m/s. In der Folge arbeiten die Simulationen mit

$n_V = n_{\mathrm{WT}} = 19$. Die FTF-Geschwindigkeit bleibt bei $v_V = 2\ m/s$.

6.4.2 Robustheit gegenüber Störungen

Zur Analyse der Robustheit der Algorithmen unter dynamischen Bedingungen testet dieser Abschnitt die Algorithmen unter zwei realitätsorientierten Störszenarien mit unterschiedlichen Stationsverfügbarkeiten:

- **Störszenario 1**: Verfügbarkeit $V_M = 0.98$, $MTTR = 600s$, $TTR \in [0, 900s]$,
- **Störszenario 2**: Verfügbarkeit $V_M = 0.90$, $MTTR = 600s$, $TTR \in [0, 900s]$.

Die Einstellung der Störungsparameter in Tecnomatix Plant Simulation erfolgt anhand der Beschreibung in Abschnitt 6.2.1. Die jeweilige $MTTR$ liegt nach Analysen der realen E-Achsproduktion bei $600s$, wobei in Tecnomatix Plant Simulation eine obere Grenze (vgl. Abschnitt 6.2.1) von $900s$ gewählt wird, weil längere Störungen unüblich sind. Die bei einer Störung anfallende „Time To Restoration“ (TTR) liegt damit im Intervall $[0, 900s]$ und basiert in der Simulation auf einer Erlang-Verteilung. Im ersten Szenario hat jede Maschine eine Verfügbarkeit von $V_M = 0.98$, was klassischen Sollvorgaben in der Automobilindustrie entspricht. Das zweite Szenario simuliert hingegen eine Produktion mit höheren Ausfallraten bei einer Verfügbarkeit von $V_M = 0.90$. Dadurch kann untersucht werden, wie die Steuerungsansätze in höher-dynamischen Systemen agieren. Die jeweilige $MTBF$ ergibt sich aus $MTTR$ und V_M, wobei die tatsächlichen Abstände zwischen Störungen einer negativen Exponentialverteilung folgen.

Die Experimente sind dabei auf die Steuerung der Default-MP aus Abschnitt 6.1 (Evaluationsbeispiel 1) ausgelegt, wobei die im letzten Abschnitt geschlussfolgerte Erweiterung auf $n_V = n_{\mathrm{WT}} = 19$ Fahrzeuge und WTs gilt. Zuerst erläutert der Abschnitt die Ergebnisse zu $V_M = 0.98$, bevor er $V_M = 0.90$ dokumentiert. Jedes Störszenario wird mit fünf unterschiedlichen Seed-Einstellungen auf Tecnomatix Plant Simulation-Seite simuliert. Zu jedem fest definierten Seed-Szenario erfolgen wiederum fünf Simulationsläufe pro Ansatz.

Verfügbarkeit von 0.98

Für die Experimente mit $V_M = 0.98$ nutzen die drei Algorithmen nach wie vor die jeweils abgeleitete Default-Konfiguration aus Abschnitt 6.3.

Tabelle 6.10 dokumentiert die Durchsatzergebnisse für die fünf Störungs-Seeds[4] pro Ansatz, wobei der Durchschnitt des eingeschwungenen Zustands ab $7200s$ dargestellt ist. Die unterschiedlichen Ergebnisse in Abhängigkeit vom Störungs-Seed zeigen die Unterschiedlichkeit der Simulationsläufe. Dabei weichen die erzielten Durchsätze aufgrund der Störungen

[4] Anhang D.2 zeigt die Varianz der Simulationsläufe der fünf unterschiedlichen Störungs-Seeds beispielhaft für den zentralen Steuerungsansatz.

vom theoretischen Optimum $D_{max} = 37.8\ 1/h$ deutlich ab. In allen Szenarien erzielt der lernende Ansatz die besten Ergebnisse, was die blaue Markierung kennzeichnet. Wie sich der zentrale und dezentrale Ansatz im Vergleich zur Heuristik schlagen, ändert sich von Szenario zu Szenario. Damit lässt sich eine klare Tendenz erkennen, dass der lernende Ansatz mit Störungen effektiver umgeht als die anderen Algorithmen sowie die Vergleichsheuristik. Nachdem die Heuristik in den statischen Szenarien meist besser agierte, erzielt der lernende Ansatz in diesem dynamischen Szenario einen um bis zu 2.2% besseren Durchsatz. Der dezentrale Ansatz agiert für vier von fünf Seed-Einstellungen besser als der zentrale.

Tabelle 6.10: Dynamische Simulationsergebnisse der drei Algorithmen im Vergleich zur eingeführten Heuristik in der Default-MP mit $n_V = n_{\text{WT}} = 19$, $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, für fünf unterschiedliche Störungs-Seeds, fünf Simulationen pro Seed.

		Zentral	**Dezentral**	**Lernend**	**Heuristik**
Nr.	Seed	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$
1	1	35.31	35.79	36.27	35.71
2	2	33.30	33.12	33.48	33.23
3	3	34.12	35.06	35.49	35.46
4	4	34.23	35.26	35.57	34.80
5	5	34.71	35.29	35.92	35.78

Für das Szenario mit Störungs-Seed vier bleibt der lernende Ansatz im Durchsatz kontinuierlich oberhalb der anderen Steuerungslösungen, wie Abbildung 6.14 zeigt. Dabei scheint der Agent die Störungen besser zu antizipieren, wodurch minimale Durchlaufzeit-Vorteile entstehen. Wiederum weist zwar der dezentrale Ansatz aus den bereits in Abschnitt 6.4.1 abgeleiteten Gründen die geringsten Durchlaufzeiten auf. Darauf folgt jedoch der lernende Ansatz mit $DZ_{J,m} = 2005.51s$ im Vergleich zur zentralen Steuerung mit $DZ_{J,m} = 2045.75s$ und zur Heuristik mit $DZ_{J,m} = 2034.50s$. Die Verteilung der Durchlaufzeiten dokumentiert Anhang D.3. Die kleineren Durchlaufzeiten führen potenziell zu höheren Durchsatzwerten.

Verfügbarkeit von 0.90

Für die Experimente mit $V_M = 0.90$ nutzen die drei Algorithmen auch hier die jeweils abgeleitete Default-Konfiguration aus Abschnitt 6.3, wobei der lernende Ansatz mit $n_{\text{env}} = 4$ parallelen Umgebungen und $n_{\text{episodes}} = 2000$ Episoden arbeitet. Die Steigerung von zwei auf vier Umgebungen sowie von 1000 auf 2000 Episoden ist hierbei notwendig, weil durch die niedrigeren Stationsverfügbarkeiten, komplexere Steuerungssituationen auftreten, für die der PPO-Agent mehr Lerndaten benötigt.

Tabelle 6.11 dokumentiert die Durchsatzergebnisse für die fünf Störungs-Seeds[5] pro Ansatz, wobei die Durchsätze dem Durchschnitt des eingeschwungenen Zustands ab $7200s$ ent-

[5] Anhang D.4 zeigt die Varianz der Simulationsläufe der fünf unterschiedlichen Störungs-Seeds beispielhaft für den dezentralen Steuerungsansatz.

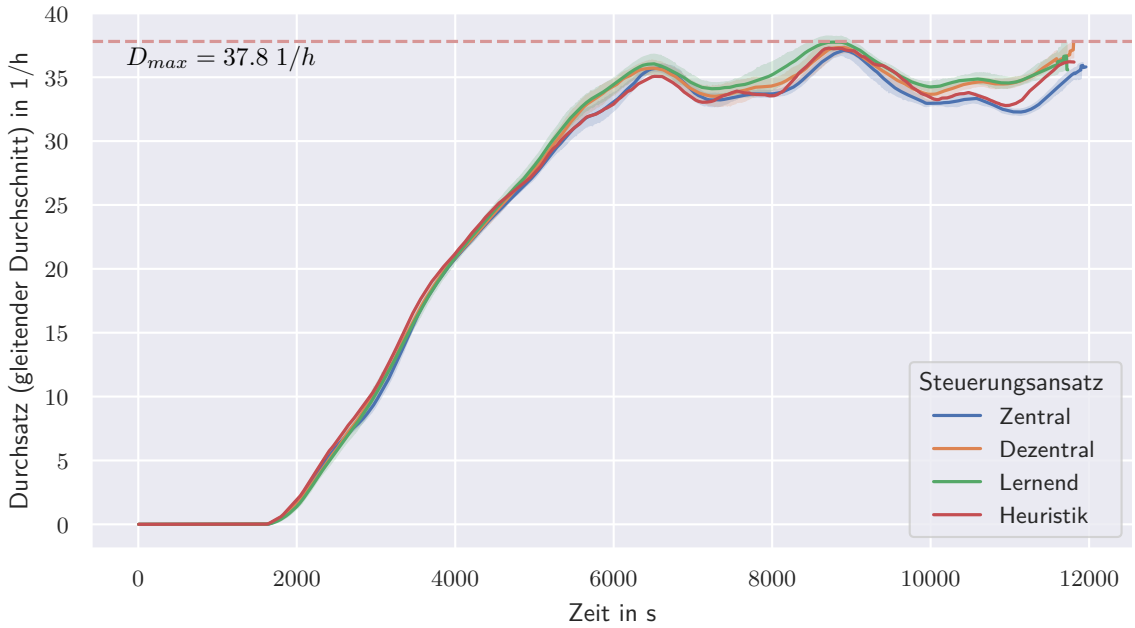


Abbildung 6.14: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed vier, fünf Simulationsläufe pro Ansatz.

sprechen. Dabei weichen die erzielten Durchsätze aufgrund der mit $V_M = 0.90$ noch häufiger auftretenden Störungen wesentlich deutlicher vom theoretischen Optimum $D_{max} = 37.8\ 1/h$ ab, als es bei $V_M = 0.98$ der Fall ist.

Tabelle 6.11: Dynamische Simulationsergebnisse der drei Algorithmen im Vergleich zur eingeführten Heuristik in der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, $V_M = 0.90$ und $MTTR = 600s$, $TTR \in [0, 900s]$, für fünf unterschiedliche Störungs-Seeds, fünf Simulationen pro Seed.

		Zentral	**Dezentral**	**Lernend**	**Heuristik**
Nr.	Seed	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$
1	1	25.00	25.86	26.93	25.36
2	2	25.82	26.33	26.66	26.30
3	3	22.54	22.24	22.88	23.00
4	4	27.39	26.40	27.39	27.32
5	5	24.71	24.88	26.21	25.63

In vier von fünf Szenarien erzielt der lernende Ansatz auch hier die besten Ergebnisse, wobei bei Seed vier der zentrale Ansatz gleich gut agiert. Für Seed drei erzielt die Heuristik ein geringfügig besseres Durchsatzergebnis. Dennoch existiert mit Seed eins auch ein Szenario, bei dem der lernende Ansatz um 6.2% besser als die Heuristik, um 4.1% besser als der dezentrale und um sogar 7.7% besser als der zentrale Ansatz abschneidet. Damit agiert der lernende Ansatz tendenziell auch bei häufigeren Störungen effektiver als die anderen

Algorithmen sowie die Vergleichsheuristik.

Für den deutlichsten Performanzunterschied bei Seed eins zeigt Abbildung 6.15 die Verteilungen der Durchlaufzeiten über alle Simulationsläufe pro Steuerungsansatz. Wiederum weist der dezentrale Ansatz mit 2405.96s aus bekannten Gründen die niedrigste mittlere Durchlaufzeit auf. Darauf folgt der lernende Ansatz (2550.25s) mit deutlichem Abstand von der Heuristik (2676.23s) und vom zentralen Ansatz (2745.53s).

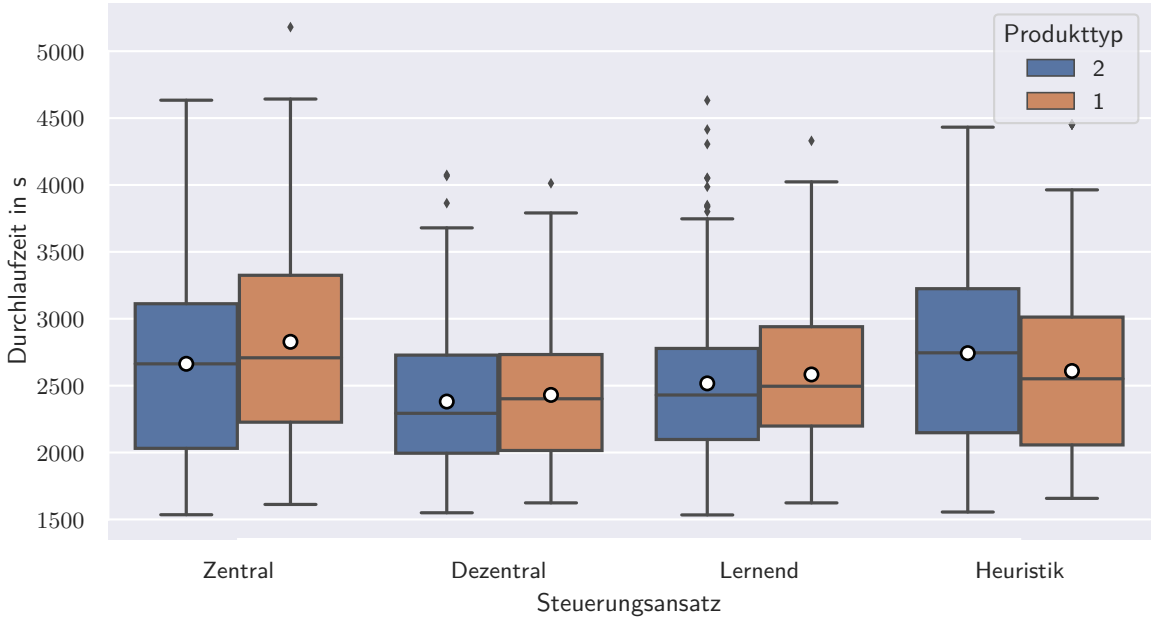


Abbildung 6.15: Durchlaufzeitverteilungen aller Steuerungsansätze über alle fünf Simulationsläufe der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.90$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins.

Außerdem lässt sich anhand des Boxplots erkennen, dass beim lernenden Ansatz der Interquartilsabstand geringer ist als beim zentralen Ansatz und bei der Heuristik. Damit liegen die mittleren 50% der Daten näher zusammen. Der PPO-Agent scheint dabei die Produkte gleichmäßiger durch die Produktion zu leiten. Insgesamt führen die wesentlich geringeren Durchlaufzeiten des Agenten, als Ergebnis des scheinbar effektiven Umgangs mit Störungen, zu besseren Durchsatzergebnissen. Wahrscheinlich nutzt der Agent hierbei die ihm im State s_t der MP zur Verfügung gestellten Daten zu aktuellen Stationsstörungen. Dabei weiß er, ob und wie lange die jeweilige Station sich bereits in Störung befindet. Diese detaillierten Störungsinformationen nutzen die anderen Ansätze in dieser detaillierten Form nicht. Gleichzeitig ist jedoch auch festzuhalten, dass der Agent nicht in sämtlichen Störszenarien um mehrere Prozent im Schnitt besser agiert als die anderen Ansätze. Potenziell lernt er also mit speziellen Produktionssituationen einen besonders effektiven Umgang. Eine detailliertere Analyse der erlernten Entscheidungsstrategien des PPO-Agenten ist aufgrund seiner

Blackbox-Natur leider erschwert, was einen Nachteil von DRL darstellt.

Die zugehörigen Durchsatzkurven dokumentiert Anhang D.5. Anhang D.6 zeigt zusätzlich die Generalisierungsfähigkeit des PPO-Agenten auf, indem die gelernten Modelle bei Störungs-Seed eins zur Steuerung der anderen Störungs-Seeds verwendet werden. Die Simulationsläufe zeigen dabei nahezu identische Durchsatzergebnisse.

6.5 Generalisierbarkeit auf erweiterte Evaluationsbeispiele

Nachdem der vorherige Abschnitt die dritte Forschungsfrage hinsichtlich des Umgangs der Algorithmen mit Störungen adressiert, befasst sich dieser Abschnitt abschließend mit der Generalisierbarkeit auf Variationen in der MP-Konfiguration. In der Realität unterliegt eine Steuerung hierbei Änderungen in der Produktion: Beispielsweise ist ein neuer Produkttyp einzurüsten, eine neue Station wird der MP hinzugefügt, es erfolgt eine Steigerung der Kapazität der Produktion als Ausbaustufe oder es verringert sich die Taktzeit an einer Station. Eine Produktionssteuerung ist deshalb darauf auszulegen, mit derartigen Herausforderungen bestmöglich umzugehen. In der Folge analysiert die Arbeit die Einführung eines dritten Produkttyps in Abschnitt 6.5.1 sowie die Integration einer neuen Station in Abschnitt 6.5.2. Für die Experimente nutzen die drei Algorithmen die Default-Parameter aus Abschnitt 6.3, wobei der lernende Ansatz mit $n_{\text{env}} = 4$ parallelen Umgebungen und $n_{\text{episodes}} = 2000$ Episoden arbeitet. Wiederum erfolgt ein Vergleich mit der in Abschnitt 6.4 eingeführten Heuristik sowie mit einem theoretischen Durchsatzoptimum.

6.5.1 Dritter Produkttyp

Die Einführung einer neuen E-Achsvariante im gegebenen Evaluationsbeispiel ist ein Szenario aus der Realität. Dabei entspricht der dritte Produkttyp, welchen bereits Tabelle 6.1 einführte, einer Variante des zweiten Produkttyps. Jedoch benötigen die ersten drei Operationen Spezialwerkzeug, welches lediglich an jeweils einer Station eingerüstet wird. Damit entfällt bei PT_3 die Flexibilität der ersten drei Operationen, worauf sich die Steuerung einzustellen hat. Die drei Produkttypen sind im 1:1:1-Produktmix zu produzieren, jedes Produkt 50 Mal. Zur Produktion stehen $n_V = n_{\text{WT}} = 19$ Fahrzeuge und WTs zur Verfügung. Alle weiteren MP-Eigenschaften entsprechen der Default-MP aus Tabelle 6.2. Aufgrund der neuen Produktkombination berechnet sich das theoretische Durchsatzoptimum D_{max} zu 36.92 $1/h$. Die resultierende MP-Konstellation wird als Evaluationsbeispiel 2 bezeichnet.

Tabelle 6.12 dokumentiert die Durchsatzergebnisse für Evaluationsbeispiel 2 für den statischen Fall (1) und den dynamischen Fall mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, mit Störungs-Seed eins (2). Der jeweilige Durchsatz entspricht dem Durchschnitt

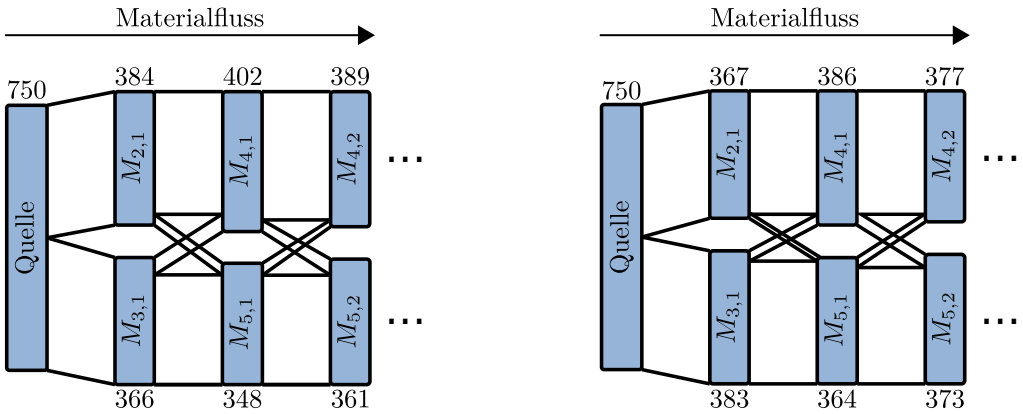
des eingeschwungenen Zustands ab 7200s über fünf Simulationen pro Ansatz.

Tabelle 6.12: Simulationsergebnisse der drei Algorithmen im Vergleich zur eingeführten Heuristik für Evaluationsbeispiel 2, fünf Simulationen pro Parameterkombination.

			Zentral	**Dezentral**	**Lernend**	**Heuristik**
Simulationen	V_M	**Seed**	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$	D_m $[1/h]$
(1) Statisch	1.0	-	36.73	35.34	36.92	36.61
(2) Dynamisch	0.98	1	33.17	32.50	34.05	32.95

Im statischen Fall erreicht lediglich der lernende Ansatz mit $D_m = 36.92$ $1/h$ das Durchsatzoptimum. Der dezentrale Ansatz fällt mit 35.34 $1/h$ deutlich ab, wobei der zentrale Ansatz mit 36.73 $1/h$ nahe am theoretischen Durchsatzoptimum liegt. Im Gegensatz zur statischen Untersuchung in Abschnitt 6.4.1 schneidet die Heuristik mit 36.61 $1/h$ nicht mehr am besten ab. Die Koordination zwischen drei Produkttypen, von denen der dritte eingeschränkt flexibel ist, bedarf höher entwickelter Strategien, die der PPO-Agent über Interaktion lernt.

Der dynamische Fall zeigt ein ähnliches Bild. Der dezentrale Ansatz fällt ab. Der lernende Ansatz erreicht 2.7% mehr Durchsatz als der zentrale. Auch hier liegt die Heuristik auf dem dritten Rang. Abbildung 6.16 analysiert die deutliche Kluft zwischen dem lernenden und dezentralen Ansatz mithilfe eines Sankey-Diagramms. Der jeweils dargestellte Materialfluss von links nach rechts für die ersten drei Operationen aller Produkte offenbart deutliche Unter-



(a) **Dezentral** (b) **Lernend**

Abbildung 6.16: Sankey-Diagramme für die Materialflüsse der fünf Simulationsläufe des dezentralen (a) und lernenden (b) Steuerungsansatzes, wobei die Zahlen die Auftragsdurchläufe pro Station repräsentieren.

schiede in der Routenwahl der Ansätze. Abbildung 6.16a zeigt für die dezentrale Steuerung, dass 384 der insgesamt 750 $(5 \cdot 150)$ Aufträge den Weg über $M_{2,1}$ beschreiten und nur 366 über $M_{3,1}$. Der lernende Ansatz zeigt hier das gegenteilige Verhältnis, wie Abbildung 6.16b darstellt. Dabei ist zu beachten, dass die erste Operation des dritten Produkttyps PT_3 ausschließlich an $M_{2,1}$ bearbeitet werden kann und eine Bearbeitungszeit von $t_{ipl} = 204s$ aufweist. Dadurch muss $M_{2,1}$ mindestens $250 \cdot 204s$ hierfür leisten. Die verbleibenden $250 \cdot 204s$ für

PT_2 sowie $250 \cdot 177s$ für PT_1 sind dabei möglichst effizient zwischen $M_{2,1}$ und $M_{3,1}$ aufzuteilen. Nur dann kann dieser Produktionsengpass optimal genutzt werden. Die Durchsatzergebnisse des lernenden Ansatzes geben Grund zu der Annahme, dass der lernende Ansatz durch seine Routenwahl eine gleichmäßigere und effektivere Engpassnutzung ermöglicht, indem er obige Zusammenhänge lernt. Demgegenüber vergibt der dezentrale Ansatz Kapazität von $M_{2,1}$, die für die unflexible erste Operation von PT_3 gebraucht würde, vermehrt an die anderen beiden Produkttypen, was zu Ineffizienzen führt. Für die weiteren Produktionsschritte unterscheidet sich das Auftragsrouting ebenfalls deutlich.

Die Betrachtung der Stationsauslastungen in Anhang E.1 bestätigt und ergänzt die obigen Zusammenhänge. Die zugehörigen Durchsatzverläufe sowie die Verteilung der Durchlaufzeiten zeigt Anhang E.2. Insgesamt agiert auch in diesem Produktionsumfeld der lernende Ansatz am besten.

6.5.2 Dritter Produkttyp und Zusatzstation

Dieser Abschnitt simuliert ein weiteres Szenario, welches in der Realität auftreten kann. Erweiternd zu Abschnitt 6.5.1 dauert die erste Operation des eingeführten dritten Produkttypen $500s$ statt $204s$, weil der Prozess wesentlich aufwendiger im Vergleich zu PT_1 und PT_2 ist. Zum Umgang mit der dadurch erhöhten Zykluszeit führt die Produktionsplanung eine zusätzliche Station $M_{1,1}$ ein, welche für die vorangegangenen Szenarien nicht im Einsatz war (vgl. Abschnitt 6.1). Durch die dritte Station verteilt sich die Prozesszeit der jeweils ersten Operation auf drei Stationen, wobei nur $M_{2,1}$ und $M_{1,1}$ die für PT_3 notwendigen Spezialwerkzeuge bieten. Die Produkte ergeben sich zu Tabelle 6.13. Die drei Produkttypen sind im 1:1:1-Produktmix zu fertigen, wobei jedes Produkt 50 Mal zu produzieren ist. Zur Produktion stehen $n_V = n_{\mathrm{WT}} = 19$ Fahrzeuge und WTs zur Verfügung. Alle weiteren MP-Eigenschaften entsprechen der Default-MP aus Tabelle 6.2. Aufgrund der neuen Produktkombination berechnet sich das theoretische Durchsatzoptimum D_{max} zu 36.78 $1/h$. Die resultierende MP-Konstellation wird als Evaluationsbeispiel 3 bezeichnet.

Tabelle 6.14 dokumentiert die Durchsatzergebnisse für Evaluationsbeispiel 3 für den statischen Fall (1) und den dynamischen Fall mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, mit Störungs-Seed eins (2). Der jeweilige Durchsatz entspricht dem Durchschnitt des eingeschwungenen Zustands ab $7200s$ über fünf Simulationen pro Ansatz.

Im statischen Fall erreicht keiner der Ansätze das Durchsatzoptimum. Mit großer Wahrscheinlichkeit führt die Durchlaufzeiterhöhung durch die Prozesszeit von $500s$ bei PT_3 zu einer höheren Anforderung an die mindestens notwendigen Fahrzeuge und WTs, um das Durchsatzoptimum erreichen zu können (vgl. Durchlaufzeiten in Anhang E.3). Mit $D_m = 36.08$ $1/h$ erzielt der lernende Ansatz jedoch weitaus bessere Ergebnisse als der zentrale, dezentrale und heuristische Ansatz. Der dezentrale Ansatz fällt mit 33.93 $1/h$ am deutlichsten ab.

Tabelle 6.13: Zu produzierende Produkttypen in Evaluationsbeispiel 3, Neuerungen blau markiert.

	Produkttyp PT_1		**Produkttyp** PT_2		**Produkttyp** PT_3	
Nr.	t_{ipl} [s]	$M_{ipl} \subseteq M$	t_{ipl} [s]	$M_{ipl} \subseteq M$	t_{ipl} [s]	$M_{ipl} \subseteq M$
1	177	$\{M_{1,1}, M_{2,1}, M_{3,1}\}$	204	$\{M_{1,1}, M_{2,1}, M_{3,1}\}$	500	$\{M_{1,1}, M_{2,1}\}$
2	120	$\{M_{4,1}, M_{5,1}\}$	120	$\{M_{4,1}, M_{5,1}\}$	120	$\{M_{4,1}\}$
3	190	$\{M_{4,2}, M_{5,2}\}$	190	$\{M_{4,2}, M_{5,2}\}$	190	$\{M_{4,2}\}$
4	60	$\{M_{2,2}, M_{3,2}\}$	60	$\{M_{2,2}, M_{3,2}\}$	60	$\{M_{2,2}, M_{3,2}\}$
5	140	$\{M_{2,3}, M_{3,3}\}$	140	$\{M_{2,3}, M_{3,3}\}$	140	$\{M_{2,3}, M_{3,3}\}$
6	49	$\{M_{2,4}, M_{3,4}\}$	49	$\{M_{2,4}, M_{3,4}\}$	49	$\{M_{2,4}, M_{3,4}\}$
7	39	$\{M_{4,3}, M_{5,3}\}$	39	$\{M_{4,3}, M_{5,3}\}$	39	$\{M_{4,3}, M_{5,3}\}$
8	40	$\{M_{3,5}\}$	82	$\{M_{4,4}, M_{5,4}\}$	82	$\{M_{4,4}, M_{5,4}\}$
9	122	$\{M_{4,4}, M_{5,4}\}$	178	$\{M_{4,5}, M_{5,5}\}$	178	$\{M_{4,5}, M_{5,5}\}$
10	137	$\{M_{4,5}, M_{5,5}\}$	40	$\{M_{3,5}\}$	40	$\{M_{3,5}\}$
11	44	$\{M_{3,5}\}$	80	$\{M_{1,5}, M_{2,5}\}$	80	$\{M_{1,5}, M_{2,5}\}$
12	155	$\{M_{1,5}, M_{2,5}\}$	72	$\{M_{2,4}, M_{3,4}\}$	72	$\{M_{2,4}, M_{3,4}\}$
13	72	$\{M_{2,4}, M_{3,4}\}$	37	$\{M_{3,5}\}$	37	$\{M_{3,5}\}$
14	0	$\{Senke\}$	0	$\{Senke\}$	0	$\{Senke\}$

Tabelle 6.14: Simulationsergebnisse der drei Algorithmen im Vergleich zur eingeführten Heuristik für Evaluationsbeispiel 3, fünf Simulationen pro Parameterkombination.

			Zentral	**Dezentral**	**Lernend**	**Heuristik**
Simulationen	V_M	**Seed**	D_m [1/h]	D_m [1/h]	D_m [1/h]	D_m [1/h]
(1) Statisch	1.0	-	35.44	33.93	36.08	35.20
(2) Dynamisch	0.98	1	32.14	31.69	33.05	32.77

Der zentrale Ansatz agiert um 0.7% besser als die Heuristik. Wie im obigen Beispiel in Abschnitt 6.5.1 verliert die Heuristik bei steigender Problemkomplexität an Lösungsgüte, nachdem sie in der statischen Untersuchung in Abschnitt 6.4.1 noch die besten Ergebnisse erzielte. Die Steigerung der Problemkomplexität zeigt sich hier anhand des Zusatzprodukts, einer erhöhten Fertigungszeitspreizung aufgrund der 500s-Operation von PT_3 sowie der zusätzlich zur Verfügung stehenden Station bei der ersten Operation der Aufträge.

Die dynamischen Simulationen zeigen ähnliche Ergebnisse, wobei die Differenz zwischen lernendem und heuristischem Ansatz weniger groß ist (0.9% vs. 2.5%) und die Heuristik bessere Ergebnisse erzielt als der zentrale Ansatz. Wieder fällt der dezentrale Ansatz am deutlichsten ab. Abbildung 6.17 stellt die Entwicklung der Produktionsbestände für einen der fünf Simulationsläufe pro Algorithmus dar. Dabei ist der maximale Produktionsbestand

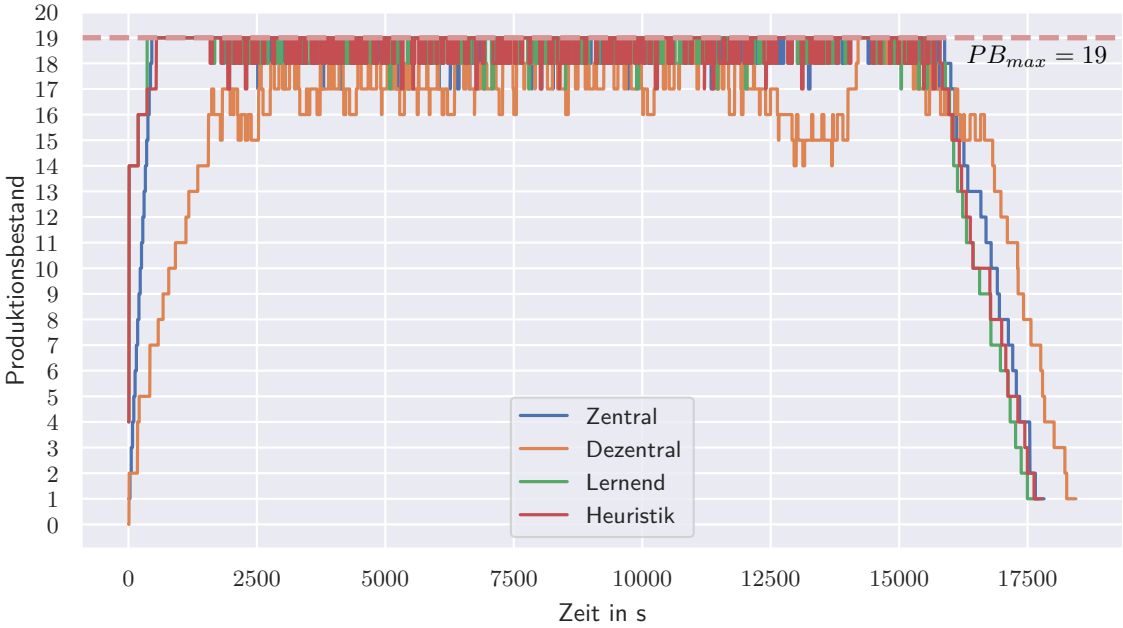


Abbildung 6.17: Entwicklung des Produktionsbestands für alle Steuerungsansätze über jeweils einen der fünf Simulationsläufe von Evaluationsbeispiel 3 mit $n_V = n_{\mathrm{WT}} = 19$, $v_V = 2\ m/s$, den Produkten aus Tabelle 6.13, $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins.

$PB_{max} = n_V = 19$ von der Anzahl der Fahrzeuge begrenzt. Wie bereits in Abschnitt 6.4.1 analysiert, baut sich der maximale Bestand für die zentrale, lernende und heuristische Steuerung schneller auf. Die orange-farbene Kurve der dezentralen Steuerung zeigt einen langsameren Bestandsaufbau. Außerdem liegt der Bestand des dezentralen Ansatzes durchgängig unterhalb der Kurven der anderen Algorithmen. Vor allem im Bereich zwischen $12000s$ und $14000s$ fällt er deutlich ab. Dieser Einbruch lässt sich auf eine Störung an Station $M_{2,1}$ zurückführen. Die Störung tritt exakt ab $12441s$ bis zur Simulationszeit von $13269s$ auf. Da Station $M_{2,1}$ Teil des Engpasses des jeweils ersten Produktionsschrittes der Produkte ist, hat dieser Ausfall direkte Auswirkungen auf den Durchsatz, wenn keine optimale Auftragseinlastung und Produktkoordination stattfindet. Vor allem Produkt PT_3 stellt hier hohe Anforderungen, weil bei Störung von $M_{2,1}$ nur noch $M_{1,1}$ für die Bearbeitung der ersten Operation infrage kommt und die zugehörige Bearbeitungszeit $500s$ beträgt. Hierfür ist ein effektives Störungsmanagement erforderlich. Der auktionsbasierte Einlastungsmechanismus des MAS führt dabei zu einer fehl-koordinierten Einlastung im Störungsfall, vor allem in Bezug zu PT_3, was den Leistungsabfall des dezentralen Ansatzes erklärt. Dass der dezentrale Ansatz auch im statischen Fall deutlich schlechter agiert, lässt ebenfalls darauf schließen, dass derartige Fehlkoordinationen bei der Einlastung auch ohne Störungen auftreten. Die zugehörigen Durchsatzverläufe sowie die Verteilung der Durchlaufzeiten zeigt Anhang E.3.

Zusammenfassend erzielt der lernende Ansatz auch in diesem Produktionsumfeld die bes-

ten Ergebnisse. Der nächste Abschnitt führt abschließend die Ergebnisse aller Experimente zusammen und leitet Handlungsempfehlungen für die Entwicklung von Steuerungsalgorithmen für die MP ab.

6.6 Zusammenfassung und Handlungsempfehlungen

Die Untersuchungen aus Abschnitt 6.3, Abschnitt 6.4 und Abschnitt 6.5 führen zu einigen Erkenntnissen, die Handlungsempfehlungen ableiten lassen. Dabei ist zu betonen, dass sich die Erkenntnisse auf die verwendeten drei Evaluationsbeispiele beziehen. Dennoch ist davon auszugehen, dass die Handlungsempfehlungen auch einen Mehrwert für alternative MP-Beispiele bieten. Zuerst fasst der Abschnitt ansatzspezifische Erkenntnisse zusammen, bevor er die Ergebnisse der vergleichenden und Generalisierungsexperimente aggregiert sowie Handlungsempfehlungen ableitet.

6.6.1 Ansatzspezifische Erkenntnisse

Für jeden der drei Steuerungsansätze ergeben sich unterschiedliche Erkenntnisse in der Konzeptionierung, Implementierung und Evaluation. Die Untersuchungen des **zentralen Steuerungsansatzes** lassen folgende Schlussfolgerungen ziehen:

- Trotz der metaheuristischen Herangehensweise sowie der zeitlichen Zerlegung des Problems mithilfe des rollierenden Schedulings ergibt sich ein Rechenzeitproblem bei der gegebenen Implementierung. Die Begrenzung der Rechenzeit pro Optimierung auf $t_{\text{opt}} = 2s$ ermöglicht trotzdem gute Steuerungsergebnisse. Dabei ist auf die prototypische Implementierung hinzuweisen. Eine zukünftige Performanzsteigerung könnte die Rechenzeitproblematik lösen.

- Für die Schedule-Berechnungen reicht eine Vorausschau um $\Delta T_{\text{GA}} = 200s$, was ungefähr zweimal der theoretischen Zykluszeit entspricht, für gute Ergebnisse aus. Schaut der GA weiter voraus, verschlechtern sich die Durchsatzergebnisse. Daher ist eine Vorausschau von ca. zweimal der theoretischen Zykluszeit t_{zyk} zu empfehlen.

- Für die Funktionsfähigkeit des Ansatzes ist die Datenaktualität in der zentralen Wissensbasis zu jeder Zeit sicherzustellen. Die realitätsnahe, allerdings simulative Implementierung erreicht dieses grundlegende Ziel. Jedoch ist davon auszugehen, dass in der Realität zusätzliche Störfaktoren wie Netzwerkausfälle oder Ausfälle der zentralen Scheduling-Einheit die Datenaktualität bzw. die Ansatzfunktionalität gefährden können. Hierfür braucht es Backup-Lösungen. Außerdem ist sicherzustellen, dass sich während einer Schedule-Berechnung der Produktionsstatus nicht ändert. Deshalb berücksichtigt jede Produktionsentität Informationen zu aktuellen Optimierungsläufen.

Die Untersuchungen des **dezentralen Steuerungsansatzes** lassen folgende Schlussfolgerungen ziehen:

- Die Vorausschau ΔT_{MAS} zum Starten von Auktionen aus Stationssicht ist so groß zu wählen, dass in der MP auftretende Transportzeiten kompensierbar sind. Schauen die Stationen nicht weit genug voraus, erhöhen die Transportzeiten die Durchlaufzeiten der Produktionsaufträge. Schauen die Stationen hingegen zu weit voraus, erschwert sich die Koordination zwischen um Kapazität konkurrierenden Produkten. Die Durchsatzergebnisse verschlechtern sich. Eine wie in Abschnitt 6.3.2 durchgeführte empirische Untersuchung ermöglicht ein Annähern und die damit verbundene Parameterwahl.

- Der wesentliche Baustein im MAS ist die erfolgreiche Kommunikation zwischen den Produktionsentitäten sowie die dezentrale Datenverfügbarkeit. Hierbei zeigt sich eine höchstmögliche Entkopplung der Agenten als förderlich, damit die Agenten nicht voneinander abhängen. Ein Mittel zur Entkopplung ist die Deadline T_{dead} bei Durchführung einer Auktion. Alle Gebote, die nicht innerhalb der Deadline bei einer Station ankommen, ignoriert die Station. Sie wartet damit nicht auf Absagen von Produkten. Das Warten auf Absagen erzeugt Abhängigkeit, weil beispielsweise auf eine ausgefallene Recheneinheit eines Produkts gewartet wird. Die Entkopplung ermöglicht auch einen besseren Umgang mit Ausfällen von Stationen zugeordneten Recheneinheiten. Im Vergleich zum zentralen Ansatz führt die Entkopplung dazu, dass der Ausfall einer Stations-Recheneinheit durch eine Alternativstation teilweise kompensiert werden kann. Das System zeichnet sich durch eine gewisse Redundanz aus. Dabei startet die rechenseitig ausgefallene Station keine Auktionen mehr, die Alternativstation schon. Fällt im zentralen Ansatz die Scheduling-Einheit aus, trifft die Steuerung hingegen keine Entscheidungen mehr.

Die Untersuchungen des **lernenden Steuerungsansatzes** lassen folgende Schlussfolgerungen ziehen:

- Action-, State- und Reward-Design sind von essenzieller Bedeutung für erfolgreiches Lernen, was die Ergebnisse von [283] und [280] bestätigt. Ein gutes Agentendesign ermöglicht dem Agenten, seine Strategien eigens an die gegebene bzw. wandelnde Problemstellung anzupassen. Dabei verfolgt er ein globales Produktionsziel, welches ihm durch die Reward-Funktion gegeben wird.

- PPO erweist sich als robust gegenüber wandelnden Problemkonfigurationen. Dabei sind wenig bis keine Parameteranpassungen notwendig. Lediglich die Anzahl der Umgebungen n_{env}, mit denen der Agent interagiert, sowie die Anzahl der Episoden n_{episodes} sind bei steigender Problemkomplexität durch z.B. häufige Störungen oder Zusatzprodukte tendenziell zu erhöhen. Dadurch sammelt der DRL-Agent in der Lernphase mehr Erfahrungen, aus denen er optimale Entscheidungen schlussfolgert.

- Action Masking steigert die PPO-Performanz hinsichtlich der Ergebnisqualität und der Lerngeschwindigkeit signifikant und lässt den Agenten auf das Wesentliche konzentrieren, was die Ergebnisse von [290], [302], [303] bestätigt.

Der folgende Abschnitt fasst die Durchsatzergebnisse der vergleichenden und generalisierenden Experimente zusammen.

6.6.2 Ergebniszusammenfassung

Die Ergebnisse aus Abschnitt 6.4 und Abschnitt 6.5 fasst Abbildung 6.18 zusammen. Evaluationsbeispiel 1 repräsentiert das Default-Beispiel aus Abschnitt 6.1. Evaluationsbeispiel 2 und 3 stellen die generalisierten Beispiele in Abschnitt 6.5.1 und Abschnitt 6.5.2 dar. Pro Beispiel sind die gemittelten Durchsatzergebnisse aller Simulationsläufe für den jeweils statischen und dynamischen Fall dargestellt.[6]

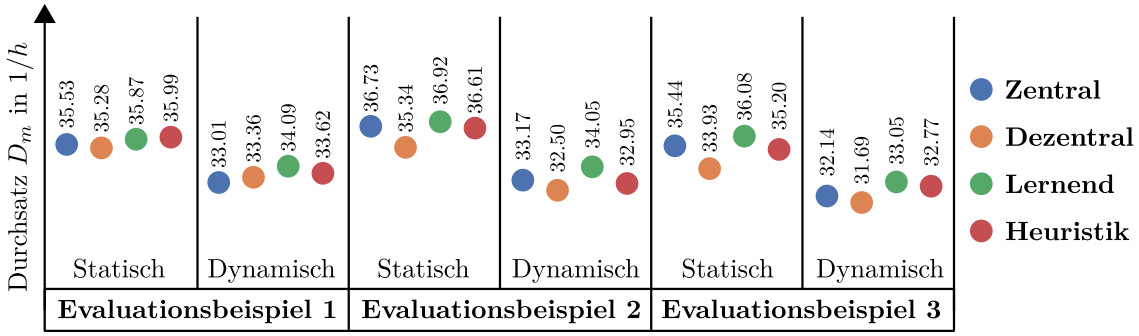


Abbildung 6.18: Gemittelte Durchsatzergebnisse aller Simulationsläufe der drei betrachteten Evaluationsbeispiele im jeweils statischen und dynamischen Simulationsfall.

Bei Betrachtung der Ergebnisse zeigt sich die hohe Lösungsgüte der lernenden Steuerung. Bis auf den statischen Fall bei Evaluationsbeispiel 1 erzielt der PPO-Agent durchgängig die besten Durchsatzergebnisse. Er agiert dabei bis zu 2.7% besser als die jeweils zweitbeste Lösung (vgl. Evaluationsbeispiel 2, dynamisch).

Der zentrale Ansatz zeigt sich für einige Konfigurationen vergleichsweise nah an der lernenden Steuerung, während der dezentrale Ansatz hinsichtlich der Lösungsgüte nahezu durchgängig abfällt. Auffällig ist auch die gute Lösungsgüte, die die Heuristik erreicht, welche in einigen Szenarien bessere Ergebnisse erzielt als die zentrale und dezentrale Steuerung. Insgesamt scheint die Durchlaufzeitminimierung in den Steuerungen erfolgreich zu einer Durchsatzmaximierung zu führen, wie das Gesetz von Little in Abschnitt 2.1.3 vorgibt.

[6] Im statischen Fall von Evaluationsbeispiel 1 gehen bei der dezentralen Steuerung auch die niedrigeren Durchsätze der drei Szenarien mit $v_V = 1\ m/s$ ein, welche durch eine zu niedrige Vorausschau bei größeren Transportzeiten zustande kommen. Für die Kombination Evaluationsbeispiel 1/Statisch ist für den dezentralen Ansatz also noch Verbesserungsspielraum durch eine höhere Vorausschau anzumerken.

Aus der Dynamik durch Stationsstörungen resultiert eine erwartete Durchsatzreduzierung. Die Verhältnisse der Lösungsgüten zwischen den Ansätzen bleiben jedoch beim Vergleich des statischen und dynamischen Falls pro Evaluationsbeispiel relativ identisch. Dabei scheint die jeweilige Produktionskonfiguration mehr Einfluss auf das verhältnismäßige Abschneiden zu haben als die Dynamik.

Die deutlichsten Aussagen, basierend auf den Ergebnissen, betreffen die lernende Steuerung und die Vergleichsheuristik:

- **Lernende Steuerung**: Die lernende Steuerung, respektive der PPO-Agent, erzielt nahezu durchgängig bessere Durchsatzergebnisse als die konkurrierenden Algorithmen.

- **Vergleichsheuristik**: Trotz ihrer Einfachheit erzielt die Heuristik in den untersuchten MP-Konfigurationen gute Durchsatzergebnisse.

Daraus lassen sich im nächsten Abschnitt Handlungsempfehlungen für die Entwicklung von MP-Steuerungssystemen ableiten.

6.6.3 Handlungsempfehlungen für die Steuerungsentwicklung

Basierend auf den dokumentierten Erkenntnissen und Ergebnissen spricht die vorliegende Arbeit zwei Handlungsempfehlungen aus, die im Folgenden begründet werden.

Handlungsempfehlung 1: Lernende Steuerung bei höherer Komplexität

In den untersuchten Evaluationsbeispielen erzielt der PPO-Agent durchgängig bessere Ergebnisse, sobald die Problemkomplexität durch Störungen, Zusatzprodukte und Zusatzstationen steigt. Daraus lässt sich die Fähigkeit des lernenden Ansatzes schließen, sich an Veränderungen der MP-Konfiguration sowie an auftretende Störungsdynamik anzupassen. Die Ergebnisse von DRL in der Spieleindustrie (vgl. z.B. [176], [177]) eröffnen außerdem Potenziale für eine weitere Erhöhung der Problemkomplexität. Dabei ist beispielsweise an eine flexiblere FTF-Flotte zu denken, welche die Produkte nicht bis zur Fertigstellung begleitet, sondern nach Durchführung eines Transports auch andere Transporte umsetzt, anstatt auf das abgelieferte Produkt zu warten. Dadurch steigt die FTF-Auslastung und die MP benötigt weniger Fahrzeuge. Die lernende Steuerung ist damit nicht nur für die in der vorliegenden Arbeit untersuchten Beispiele vielversprechend, sondern bietet zusätzlich Spielraum für zukünftige Flexibilisierungsideen. Außerdem ermöglicht die Adaptionsflexibilität eine effiziente Anpassung an rekonfigurierbare Produktionssysteme. Diese Arbeit empfiehlt deshalb die Entwicklung lernender Steuerungssysteme für die MP für höhere Komplexitätsgrade, wobei zu klären ist, wie sich eine simulativ angelernte Steuerung konkret in eine reale Produktion überführen lässt.

Handlungsempfehlung 2: Verzahnung von Planung und Steuerung

Ist die Komplexität des Steuerungsproblems eingeschränkt, wie beispielsweise im statischen Fall des ersten Evaluationsbeispiels, reicht die Einfachheit der verwendeten Vergleichsheuristik für eine effektive Steuerung aus. Sie schneidet für diesen Fall sogar geringfügig besser ab als die lernende Steuerung. Auch in den komplexeren MP-Konfigurationen erzielt die Heuristik gute Ergebnisse. Dies lässt darauf schließen, dass eine wenig komplexe Produktion keinen aufwendiger zu implementierenden PPO-Agenten benötigt, sondern eine einfache Heuristik ausreichen kann. Befindet sich ein Produktionssystem bereits in der Betriebsphase (Brownfield-Anwendung), sind hierbei die Gegebenheiten für die Steuerung bereits klar definiert. Dabei ist eine anwendungsindividuelle Untersuchung empfehlenswert, welche die Grenze einer einfachen Heuristik ermittelt und analysiert, ob sich die Entwicklung einer komplexeren DRL-Steuerung finanziell lohnt. Bei Planung einer neuen MP (Greenfield-Anwendung) ist hingegen eine verzahnte Planung und Steuerung zu empfehlen. Dabei ist monetär zu bewerten, ob sich die jeweilige Planungs-/Steuerungskonfiguration auszahlt. Möglicherweise lässt sich eine MP planen, in der eine einfache Heuristik zur Steuerung ausreicht. Andernfalls könnten auch die Effizienzvorteile einer DRL-Steuerung dazu führen, dass in der Produktionskonfiguration gespart wird. In jedem Greenfield-Fall ist die Verzahnung von Planung und Steuerung deshalb empfehlenswert, wobei die Planung hier die Produktionsgestaltung meint.

Nach Erreichung des Ziels aus Abschnitt 1.2 nach Handlungsempfehlungen für die Steuerungsentwicklung in der MP zieht das nächste Kapitel ein Gesamtfazit und dokumentiert zukünftige Handlungsfelder.

7 Fazit und Ausblick

Die Produktionsindustrie befindet sich in der vierten industriellen Revolution. Dabei verfolgt die deutsche Regierung die Ziele einer Produktindividualisierung und Produktionsflexibilisierung. Gleichzeitig bringt die steigende Variantenvielfalt, die neben der Individualisierung auch von der gesetzlich getriebenen Entwicklung neuer Antriebskonzepte geprägt ist, die Produktionseffizienz von Fließfertigungssystemen in der Automobilindustrie an ihre Grenzen. Daraus entwickelte sich das Produktionsparadigma der MP mit entkoppelt angeordneten Stationen und einem flexiblen Materialtransport durch ein FTS. Zur Nutzung der Effizienzvorteile der MP herrscht Handlungsbedarf in der Entwicklung effektiver Steuerungssysteme. Hierbei erfordert eine derartige Steuerung eine Koordination der Auftragseinlastung, ein effektives Stationsmanagement und Auftragsrouting, eine Koordination der Fahrzeugflotte in Form eines Flottenmanagements sowie einen robusten Umgang mit Störungen durch ein Störungsmanagement.

Dabei stellt sich die vorliegende Arbeit drei Forschungsfragen mit dem Ziel, realistische Handlungsempfehlungen für die Umsetzung einer geeigneten Steuerung unter variierenden Produktionskonfigurationen und in Abhängigkeit der Beschaffenheit von Ressourcenstörungen zu geben. Die folgenden Absätze beantworten die gestellten Forschungsfragen.

1. Forschungsfrage: Welche Steuerungsarchitekturen für die MP stehen im Stand der Forschung zur Verfügung?

Das Kapitel 2 führt die Grundlagen zum Verständnis der vorliegenden Arbeit ein. Dabei untersuchen Abschnitt 2.2.4, Abschnitt 2.3.2 und Abschnitt 2.3.3 zur Verfügung stehende Steuerungsparadigmen. Über die Jahre entwickelten sich zentrale, hierarchische, modifiziert hierarchische sowie heterarchische Architekturen. Die zentrale Architektur arbeitet mit einer einzigen Entscheidungsentität. Ab der hierarchischen Architektur treffen auch andere Entitäten zumindest lokal Entscheidungen. In heterarchischen Strukturen arbeiten Entitäten vollständig autonom. Die vorliegende Arbeit betrachtet hierbei hierarchische, modifiziert hierarchische sowie heterarchische Architekturen als dezentral, weil zumindest teilweise verteilt entschieden wird. Neben der zentralen und dezentralen Entscheidungsfindung erarbeitet die Dissertation die lernende Entscheidungsfindung als geeignetes Paradigma, welches vor allem im Kontext von Industrie 4.0 an Relevanz gewinnt. Um das Ziel realistischer Handlungsempfehlungen für die MP zu erreichen, entwickelt und untersucht die vorliegende Arbeit, der Steuerungsevolution folgend, eine zentrale, eine dezentrale und eine lernende Steuerung.

2. Forschungsfrage: Wie lassen sich geeignete Steuerungsarchitekturen in der MP umsetzen, um eine gegen Ressourcenstörungen robuste, in Bezug auf den Durchsatz möglichst optimale sowie realitätsnahe Steuerung zu konzipieren?
Zur Ableitung von Handlungsempfehlungen für die MP-Steuerung entwickelt die vorliegende Arbeit einen zentralen (vgl. Kapitel 3), einen dezentralen (vgl. Kapitel 4) und einen lernenden (vgl. Kapitel 5) Steuerungsansatz. Die Konzeption der Algorithmen orientiert sich an Steuerungen verwandter Produktionsarten wie der Werkstattfertigung und flexiblen Fertigungssystemen. Die analysierten Ansätze bilden hierbei eine solide Grundlage für die Steuerungsentwicklung, erfüllen allerdings nicht alle Anforderungen, die eine MP-Steuerung stellt. Insgesamt sind in der vorliegenden Arbeit neun Anforderungen zu erfüllen (vgl. Abschnitt 2.4.2). Zum einen sind die fünf Entscheidungskategorien der Auftragseinlastung, des Stationsmanagements, des Auftragsroutings, des Flottenmanagements und des Störungsmanagements zu lösen. Zum anderen strebt die MP-Steuerung eine Durchsatzmaximierung an, ist echtzeitfähig zu gestalten, benötigt die für die Steuerungsentscheidungen relevanten Daten auf verteilten Rechenentitäten und ist generalisierbar auf unterschiedliche Produktionskonfigurationen der MP zu konzipieren. Tabelle 7.1 stellt die Anforderungen hierzu tabellarisch dar und ordnet die drei entwickelten Ansätze ein. Dabei erfüllen die drei Steuerungen die Anforderungen mit zwei geringfügigen Defiziten. Der zentrale Ansatz obliegt Einschränkungen in der Echtzeitfähigkeit, weil die Schedule-Berechnung mit einem Zeitlimit von $2s$ arbeitet, um einen Kompromiss aus Lösungsgüte und Rechenzeit zu schließen. Dadurch verletzt er die Echtzeitanforderung von $T_{max} = 1s$ (vgl. Abschnitt 2.4.2). Jedoch beeinflussen die $2s$ Rechenzeit die Produktion nicht merklich negativ, weshalb die Echtzeitfähigkeit als „teilweise berücksichtigt“ einzustufen ist. Der lernende Ansatz zeigt ein Defizit bei der Datenverfügbarkeit, da sich die Implementierung auf die Performanz in der Lernphase konzentriert und deshalb nicht verteilt arbeitet. Jedoch ist davon auszugehen, dass die Datenverfügbarkeit in Analogie zu den Umsetzungen der anderen beiden Ansätze zukünftig ebenso sichergestellt werden kann. Deshalb erfolgt auch hier die Einordnung als „teilweise berücksichtigt“.

Die drei Steuerungsansätze beantworten die zweite Forschungsfrage individuell. Sie unterscheiden sich signifikant in der Entscheidungsfindung: Beim zentralen Ansatz berechnet eine zentrale Recheneinheit mithilfe eines GAs globale Ablaufpläne für das Produktionssystem. Die Schedule-Berechnung wird zeitlich rollierend und ereignisgesteuert angestoßen. Beim dezentralen Ansatz verhandeln Produkte mit Stationen um die Ressourcenallokation in einem auktionsbasierten MAS, welches das CNP und Erstpreisauktionen nutzt. Dabei prüfen Stationen kontinuierlich, ob sie zukünftig Kapazität anzubieten haben. Der lernende Ansatz agiert mittels „trial-and-error“ und sammelt dadurch in einer Lernphase Erfahrungen, die er später zur Steuerung nutzt. Hierbei kommt ein PPO-Agent aus dem Bereich des DRLs zum Einsatz. Die exakten Umsetzungen und damit die Anforderungserfüllung dokumentieren Kapitel 3, Kapitel 4 und Kapitel 5, ergänzt durch die Beschreibung der Implementierung in Abschnitt 6.2. Zusätzliche ansatzspezifische Erkenntnisse dokumentiert Abschnitt 6.6.1.

Tabelle 7.1: Anforderungsabgleich des zentralen, dezentralen und lernenden Steuerungskonzepts.

Legende: ● berücksichtigt ◐ teilweise berücksichtigt ○ nicht berücksichtigt, nicht ersichtlich	1.1 **Auftragseinlastung**	1.2 **Stationsmanagement**	1.3 **Auftragsrouting**	1.4 **Flottenmanagement**	1.5 **Störungsmanagement**	2 **Durchsatzmaximierung**	3 **Echtzeitfähigkeit**	4 **Datenverfügbarkeit**	5 **Generalisierbarkeit**
Rollierendes Scheduling	●	●	●	●	●	●	◐	●	●
Auktionsbasiertes Multiagentensystem	●	●	●	●	●	●	●	●	●
Deep Reinforcement Learning-Agent	●	●	●	●	●	●	●	◐	●

3. Forschungsfrage: Welche Steuerungsarchitektur ist für die MP in Abhängigkeit der Produktionskonfiguration sowie der Beschaffenheit von Ressourcenstörungen am besten geeignet?

Als Grundlage für die vergleichenden Untersuchungen der entwickelten Steuerungssysteme dient eine realitätsnahe Implementierung auf Basis der VIBN (vgl. Abschnitt 6.2). Die Implementierung setzt dabei drei Evaluationsbeispiele im Rahmen der in der Arbeit untersuchten Systemgrenze (vgl. Abschnitt 2.4.2) um, welche sich an einer E-Achsproduktion der Audi AG orientieren. Sie ermöglicht eine Evaluation unter identischen Rahmenbedingungen für jede simulierte Parameterkombination. Abschnitt 6.4 untersucht die Steuerungsalgorithmen in Abhängigkeit der Beschaffenheit von Ressourcenstörungen. Abschnitt 6.5 vergleicht hingegen die Steuerungsergebnisse in Abhängigkeit der Produktionskonfiguration. Die Ergebnisse der drei entwickelten Steuerungsansätze misst die Arbeit im Verhältnis zu einem theoretischen Durchsatzoptimum sowie einer Vergleichsheuristik. Die Heuristik entscheidet sich bei jedem Produktionsschritt für den zeitlich kürzeren Weg, wobei Transportzeiten zu Stationen sowie Warteschlangen an Stationen Berücksichtigung finden. Aus den Untersuchungen stechen zwei Ergebnisse heraus: Zum einen erzielt die lernende Steuerung nahezu durchgängig bessere Durchsatzergebnisse als die konkurrierenden Algorithmen und die Heuristik. Im Durchschnitt agiert der DRL-Agent hierbei um bis zu 2.7%, in Einzelfällen sogar um bis zu 4.1% besser als der jeweils zweitbeste Ansatz. Dabei lernt der Agent anhand seines Action-, State- und Reward-Designs effektive Strategien für alle Produktionsumgebungen. Zum anderen besticht die Vergleichsheuristik trotz ihrer Einfachheit durch gute Durchsatzergebnisse, vor allem in weniger komplexen Produktionskonfigurationen. Insgesamt scheint die jeweilige Produktionskonfiguration dabei mehr Einfluss auf das relative Algorithmus-Abschneiden zu haben als die Stationsstörungen. Abschnitt 6.6.2 fasst diese Ergebnisse zusammen.

Die Ergebnisse münden in zwei Handlungsempfehlungen, welche Abschnitt 6.6.3 dokumentiert. Erstens empfiehlt die vorliegende Arbeit die Entwicklung lernender Steuerungssysteme für die MP bei höheren Komplexitätsgraden. Dabei bietet die Methode Spielraum für eine Lockerung von in Abschnitt 2.4.2 getroffenen Annahmen (z.B. Berücksichtigung von FTF-Ausfällen), eine Vergrößerung der Produktionskonfiguration (z.B. mehr Stationen und Fahrzeuge) sowie eine ausgedehnte Flexibilisierung der Produktgraphen (z.B. Berücksichtigung von Reihenfolgeflexibilität). Zweitens ist eine Verzahnung von Planung und Steuerung bei der Umsetzung neuer modularer Produktionssysteme von elementarer Bedeutung. Arbeiten Planung und Steuerung in einer frühen Phase zusammen, lassen sich unterschiedliche Planungs-/Steuerungskonfigurationen bewerten. Möglicherweise ist eine einfache heuristische Steuerung, wie die in dieser Arbeit verwendete Vergleichsheuristik, für ein entsprechend wenig komplex gestaltetes Produktionssystem ausreichend. Die individuelle Ermittlung einer Komplexitätsgrenze, ab der sich die lernende Steuerung finanziell lohnt, ist hierbei empfehlenswert. Möglicherweise führen auch die Effizienzvorteile durch eine DRL-Steuerung zu Kosteneinsparungen in der Produktionskonfiguration.

Zusammenfassend lässt sich festhalten, dass diese Arbeit drei unterschiedliche Steuerungsalgorithmen vorstellt, welche das Potenzial einer erfolgreichen Produktionssteuerung für eine MP in der Automobilindustrie bieten. Die zentrale und dezentrale Steuerung erfordern hierfür jedoch weitere Optimierungen, um die Lösungsgüte des lernenden Ansatzes zu erreichen. Im zentralen Ansatz ist vor allem das Rechenzeitproblem zu lösen. Der dezentrale Ansatz birgt hingegen beispielsweise Verbesserungspotenzial in der Auktionsgestaltung. Die vielversprechendste Steuerung repräsentiert daher der lernende Ansatz, der PPO zum Lernen effektiver Steuerungsentscheidungen nutzt. PPO verspricht Robustheit im Umgang mit Störungen sowie Robustheit im Umgang mit variierenden Umgebungen. Das Lernen adaptierter Strategien erfordert hierbei nahezu keine Parameteranpassungen, sobald eine stabile Parameterkonfiguration gefunden ist. Im Gegensatz dazu beeinflusst eine Änderung in der Problembeschaffenheit, wie eine Erhöhung der Anzahl an Fahrzeugen, beispielsweise direkt die Rechenzeit des zentralen Ansatzes, was dessen Echtzeitfähigkeit gefährdet. Die entwickelten Algorithmen sowie die damit verbundenen Ergebnisse, Erkenntnisse und Handlungsempfehlungen leisten einen wesentlichen Beitrag zur erfolgreichen Planung, Steuerung und Umsetzung einer MP.

Ausblick

Zukünftig ist die Erweiterung des lernenden Steuerungskonzepts aufgrund der zahlreichen Vorteile gegenüber den Alternativsteuerungen zu empfehlen. Dabei kann durch Vergleiche unterschiedlicher Action-, State- und Reward-Designs in Analogie zu [283] die Lösungsgüte weiter erhöht sowie das lernende Framework auf alternative Arten der MP erweitert werden. Außerdem ist zu entwickeln, wie eine simulativ trainierte Steuerung konkret in eine reale Produktion überführt werden kann. Die damit einhergehende Herausforderung betrifft die

Frage nach der Modellierung der realen Produktion als Umgebung für den Lernprozess und wie genau bzw. abstrahiert diese Modellierung durchzuführen ist. In der realen Produktion treten auf der Ebene der Produktionskoordination mitunter Herausforderungen auf, die simulativ schwer zu erfassen sind (z.B. Werker blockieren Fahrstraßen). Daher ist die Frage zu klären, ob die Simulation die Realität ausreichend gut erfasst, damit eine erfolgreiche Inbetriebnahme der angelernten Steuerung möglich ist. Zusätzlich ist zu beweisen, dass einem lernenden Agenten in der realen Produktion die notwendigen Produktionsdaten im Sinne der Datenverfügbarkeit in Tabelle 7.1 in Echtzeit als Input zur Verfügung gestellt werden können. Außerdem gilt eine Umsetzung eines lernenden MAS als vielversprechend, was die lernende mit der dezentralen Steuerungswelt kombiniert. Dadurch verliert die Steuerung zwar teilweise die zentrale Problemsicht. Jedoch haben sich DRL-Lösungen mit mehreren Agenten bereits erfolgreich in der Literatur gezeigt (vgl. z.B. [36], [285], [288]). Die Multiagentensicht brächte den Vorteil, dass ein Ausfall einer Recheneinheit nicht die gesamte Produktion lahmlegt, sondern die funktionierenden Agenten weiter agieren. Schließlich sind die verantwortlichen Produktionsmitarbeiter in einer MP zu befähigen, eine lernende Steuerung zu bedienen, zu analysieren und notwendige Anpassungen, beispielsweise in den Parametern, durchzuführen. Alternativ ist eine autonome Parametrierung vorzusehen, sodass keine Eingriffe in der laufenden Produktion notwendig sind.

Zur Umsetzung modularer Produktionssysteme bietet sich außerdem die Entwicklung einer Methodik an, welche Planung und Steuerung im Sinne der zweiten Handlungsempfehlung verzahnt. Ein Baukasten, in welchem verschiedene Arten der MP mit unterschiedlichen Steuerungssystemen kombiniert und getestet werden können, würde diese Lücke schließen. Damit ließe sich eine optimale Konfiguration aus Produktionsgestaltung und Steuerung schlussfolgern. Durch diese Planungsgrundlage wäre der Automobilindustrie auch bei der Frage geholfen, für welche Bauteile eine MP Effizienzvorteile bringt und für welche eine andere Produktionsart vorteilhaft ist. Eine Kombination der Inhalte der vorliegenden Arbeit mit Inhalten, die den Fokus auf die Planung einer MP legen (vgl. z.B. [18]), stellt hierfür die notwendige Basis zur Verfügung. Zusätzlich bietet sich die Integration von „Smart Factory“-Technologien (vgl. z.B. [127]) an, um zukunftsfähige modulare Produktionssysteme zu entwerfen.

A Grundlagen

A.1 Framework zur Virtuellen Inbetriebnahme

In [37] wurde das in Abbildung A.1 dargestellte Framework zur VIBN auf Basis des dreiteiligen Vorgehens von Zäh et al. [188] entwickelt. Ein konfigurierbares Simulationsmodell in der DES-Software Tecnomatix Plant Simulation simuliert hierbei die Produktionsabläufe der MP. Stationen können in einer Rasterstruktur angeordnet und auf beliebige Stationszahlen skaliert werden. Dies ermöglicht die Simulation verschieden konfigurierter Produktionssysteme. Die Fahrzeuge bewegen sich auf Zweibahnstraßen. Über eine TCP/IP-Schnittstelle tauscht das Simulationsmodell Informationen mit der Steuerungslogik aus.

Beispielsweise werden Entscheidungen angefordert (Informationsaustausch), Stationsstörungen kommuniziert (einseitig informativ) oder es erfolgt ein Update der Wissensbasis eines Entscheidungsträgers der Steuerungslogik (einseitig informativ). Die Steuerungslogik wurde so implementiert, dass einerseits zentrale, aber auch dezentrale Algorithmen implementiert und getestet werden können. Hierzu arbeitet das Framework mit vier Agententypen, aus denen die Steuerungslogik zusammengesetzt werden kann. Ein Maschinen-Agent repräsentiert eine Maschine, ein Werkstückträger-Agent einen WT und ein Fahrzeug-Agent ein FTF, sie stellen also Produktionsentitäten steuerungsseitig dar. Werkstückträger-Agenten repräsentieren indirekt Produkte, da ein Produkt auf einem WT durch das Produktionssystem geleitet wird. Koordinations-Agenten finden hingegen für zentral-koordinative Aufgaben Anwendung. Ein Koordinations-Agent wird beispielsweise auf Steuerungsseite dafür eingesetzt, die TCP-Schnittstelle zum Simulationsmodell zu bedienen. Jeder Agententyp ist mit einer Wissensbasis im JSON-Format ausgestattet, wo beispielsweise der aktuelle Produktionsstatus eines Maschinen-Agenten und seine Fähigkeiten hinterlegt sind. Durch die Agentenbauweise der Steuerungslogik lässt sich neben dem Produktionssystem auch die Steuerungslogik konfigurieren, wobei jeder Agent einer eigenen Recheneinheit zugewiesen wird, um realitätsnah zu arbeiten. Die Agentenkommunikation nutzt MQTT.

Mit diesen Bestandteilen ermöglicht das Framework analog zu [189] die Implementierung zentraler Steuerungsansätze, indem beispielsweise eine aktive zentrale Scheduling-Einheit vom Agententyp Koordinations-Agent das Wissen über die Produktion nutzt, um zentral Schedules zu erstellen. Die getroffenen Entscheidungen können dann an die passiv agierenden Werkstückträger-Agenten, Maschinen-Agenten und Fahrzeug-Agenten gesendet wer-

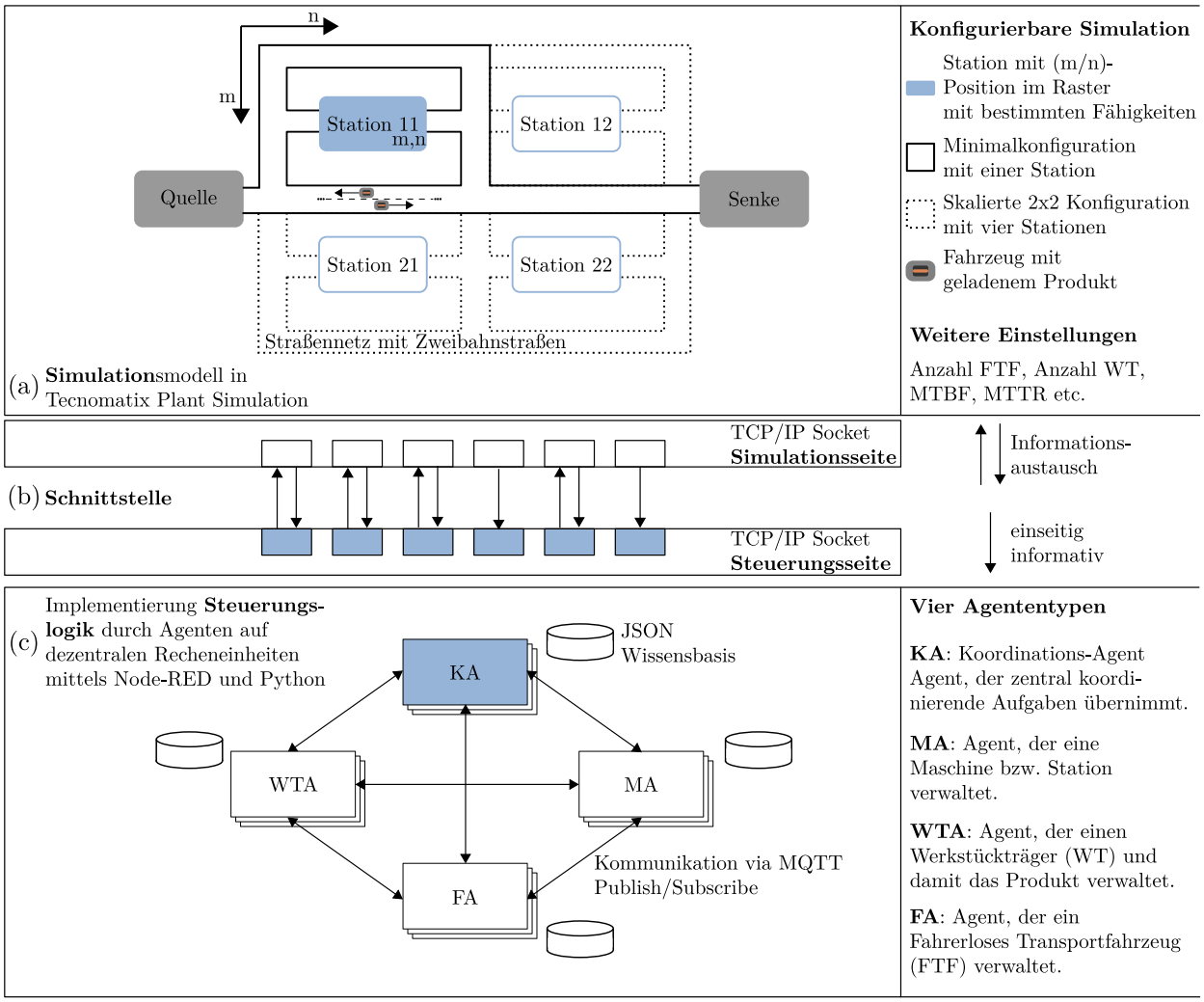


Abbildung A.1: Framework zur VIBN zentraler und dezentraler Steuerungsalgorithmen mit einem konfigurierbaren Simulationsmodell in Tecnomatix Plant Simulation (a), einer TCP/IP-Schnittstelle zur Kommunikation mit der Steuerung (b) und einer mittels vier Agententypen konfigurierbaren Steuerungslogik (c), welche in Node-RED und Python implementiert wurde, Quelle: Eigene Darstellung in Anlehnung an [37].

den. Fordert die Simulation dann eine Entscheidung für ein Produkt auf einem bestimmten WT an, wird diese Anfrage an den entsprechenden Werkstückträger-Agenten weitergeleitet, beantwortet und die Antwort an das Simulationsmodell zurückgesendet. Hingegen könnte ein dezentraler Steuerungsfall beispielsweise so aussehen, dass Werkstückträger-Agenten, Maschinen-Agenten und Fahrzeug-Agenten um die Produktionskapazität verhandeln, damit aktiv sind und nicht von einer zentralen Scheduling-Einheit abhängen. Da lernende Ansätze entweder zentral oder dezentral agieren, lässt sich das Framework auch für die Implementierung lernender Steuerungen erweitern.

B Implementierung

B.1 Verwaltetes Wissen im zentralen Ansatz

Das auf Steuerungsseite verwaltete zentrale Produktionswissen der **Scheduling-Einheit** betrifft alle Stationen, Fahrzeuge, WTs/Produkte, Störungen sowie GA-Parameter:

- Zu Stationen: Aktuell geplante Operationen mit Start- und Endzeiten, anzubietende Fähigkeiten, aktuell bearbeiteter Prozess, Verfügbarkeit ja/nein, Inhalt vom EP, Inhalt vom AP, Mindest-Transportzeiten in Puffern ($t_{\mathrm{EP},min}$, $t_{\mathrm{AP},min}$), Transportzeiten zwischen Stationen.
- Zu Fahrzeugen: Aktuell geplante Fahrten mit Start- und Endzeiten, geladener WT, aktuell durchgeführter Transport, Ort des Fahrzeugs, Geschwindigkeit v_V.
- Zu WTs/Produkten: Zugewiesenes Produkt, Auftrags-ID, aktuell geplante Operationen mit Start- und Endzeiten, durchzuführende Operationen mit Bearbeitungszeiten und benötigten Fähigkeiten, bereits abgeschlossene Operationen, aktuell anstehende Operation, aktueller Ort, betroffen von Stationsstörung ja/nein.
- Zu Störungen: Übersicht zu aktuellen Stationsstörungen mit Störungsstart $t_{\mathrm{stoer,start}}$.
- GA-Parameter: Vorausschau ΔT_{GA}, Populationsgröße N_{POP}, Anzahl selektierter Individuen N_{SEL}, Anzahl Generationen N_{GEN}, Rekombinationswahrscheinlichkeit p_R, Mutationswahrscheinlichkeit p_M, Zeit für Optimierung t_{opt}.
- Aktuelle Simulationszeit: Synchronisation der Informationen.

Die **koordinative VM**, welche u.a. für die Auftragseinlastung zuständig ist, benötigt hingegen Wissen zu verfügbaren Fahrzeugen und WTs zur Auftragseinlastung, zum Mindestabstand ΔT_{AE} zwischen zwei Einlastungen, zum Produktmix und Produktionsprogramm sowie zur aktuellen Simulationszeit.

Obige Produktionsinformationen geben der zentralen **Scheduling-Einheit** und der **koordinativen VM** einen Überblick zum aktuellen Produktionsstatus und zur Auftragseinlastung. Sie betreffen die Informationen, die zur Umsetzung des in Abschnitt 3.3 entwickelten zentralen Steuerungskonzepts notwendig sind.

B.2 Verwaltetes Wissen im dezentralen Ansatz

Das auf Steuerungsseite in den unabhängigen Node-RED-Instanzen verwaltete dezentrale Produktionswissen betrifft die **M-Agenten, V-Agenten, P-Agenten** und die **koordinative VM**, welche auch die Auftragseinlastung koordiniert:

- **M-Agent**: Vorausschau ΔT_{MAS}, aktuell geplante Operationen mit Start- und Endzeiten, anzubietende Fähigkeiten, aktuell bearbeiteter Prozess, Verfügbarkeit ja/nein, Störungsstart $t_{\text{stoer,start}}$ falls nein, Inhalt vom EP, Inhalt vom AP, Bietberechtigung für Aufträge, laufende Auktion ja/nein, aktuelle Simulationszeit.

- **V-Agent**: Aktuell geplante Fahrten mit Start- und Endzeiten, geladener WT, aktuell durchgeführter Transport, Ort des Fahrzeugs, Geschwindigkeit v_V, aktuelle Simulationszeit.

- **P-Agent**: Zugewiesenes Produkt, Auftrags-ID, aktuell geplante Operationen mit Start- und Endzeiten, durchzuführende Operationen mit Bearbeitungszeiten und benötigten Fähigkeiten, bereits abgeschlossene Operationen, aktuell anstehende Operation, aktueller Ort, betroffen von Stationsstörung ja/nein, Start der Stationsstörung falls ja, Wartezeit in AP $t_{\text{warte},i}$, Bietberechtigung ja/nein, zugewiesenes FTF, Auftragseinlastungszeitpunkt S_{J_i}, Mindest-Transportzeiten in Puffern ($t_{\text{EP},min}$, $t_{\text{AP},min}$), Transportzeiten zwischen Stationen, Stationsfähigkeiten, aktuelle Simulationszeit.

- **Koordinative VM/AE-Agent**: Verfügbare Fahrzeuge und WTs zur Auftragseinlastung, Produktmix und Produktionsprogramm, aktuelle Simulationszeit.

Obige Produktionsinformationen betreffen die dezentralen Informationen, die zur Umsetzung des in Abschnitt 4.3 entwickelten dezentralen Steuerungskonzepts notwendig sind.

C Ansatzspezifische Experimente

C.1 Zentral: Auslegung des Ermittlungszeitraums mit Nullzeitannahme

Abbildung C.1 dokumentiert für $\Delta T_{\mathrm{GA}} \in \{100s, 150s, 200s, 250s, 300s\}$ im zentralen Ansatz die durchschnittlichen Simulationsergebnisse gemittelt über jeweils fünf Simulationsläufe pro Parameterkombination. Gesteuert wird die Default-MP aus Tabelle 6.2. Dabei produziert die MP 50 Mal PT_1 und 50 Mal PT_2 im 1:1-Produktmix. Stationsstörungen bleiben aus. Es gilt die Annahme, dass die Rechenzeit unberücksichtigt bleibt, was die Anforderung an Echtzeitfähigkeit missachtet. Bei einer Analyse der Durchsatzdurchschnitte ab $7200s$ ergeben sich die Werte pro ΔT_{GA} in Tabelle C.1.

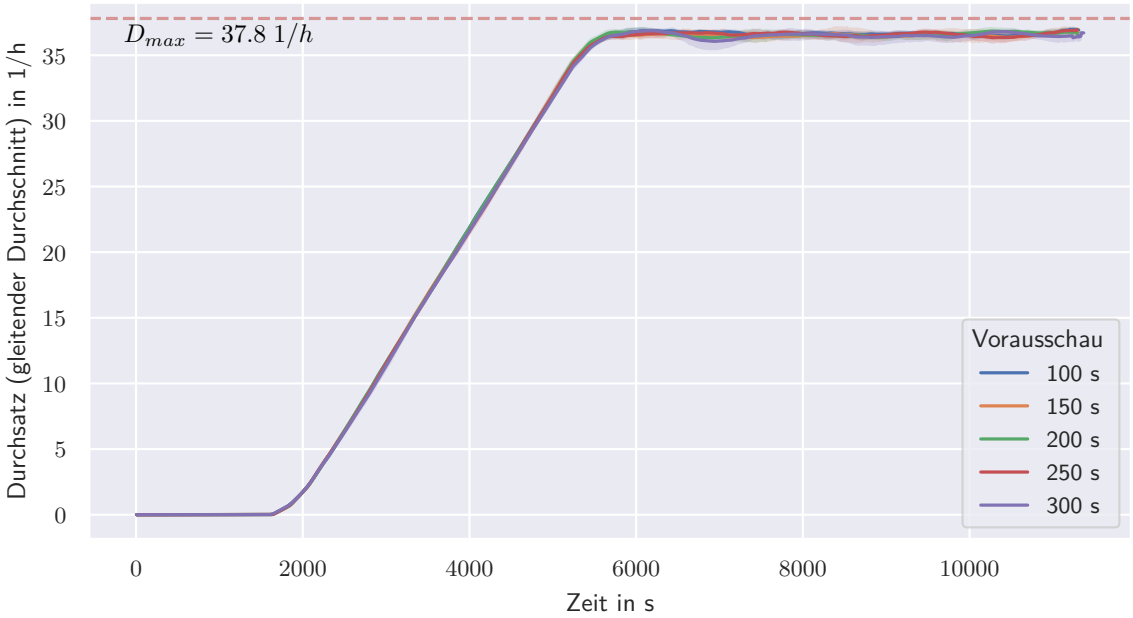


Abbildung C.1: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP aus Tabelle 6.2 mit 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, für unterschiedliche Vorausschauparameter ΔT_{GA}, fünf Simulationen pro ΔT_{GA}, wobei t_{opt} ignoriert wird und nicht in die Simulation einfließt.

Tabelle C.1: Durchschnittliche Durchsätze D_m im eingeschwungenen Zustand für die Simulationszeit ab $7200s$ in Abhängigkeit der Vorausschau ΔT_{GA}, fünf Simulationsläufe pro ΔT_{GA}.

Vorausschau ΔT_{GA}	**Scheduling-Abstand** Δt_s	**Durchsatzdurchschnitt** D_m
$100s$	$50s$	$36.63\ 1/h$
$150s$	$100s$	$36.54\ 1/h$
$200s$	$150s$	$36.61\ 1/h$
$250s$	$200s$	$36.61\ 1/h$
$300s$	$250s$	$36.54\ 1/h$

C.2 Zentral: 30 im Vergleich zu fünf Simulationen

Abbildung C.2 vergleicht die Unterschiede von 30 zu fünf durchgeführten Simulationen im zentralen Steuerungsansatz anhand der Default-Steuerung mit den Parametern aus Tabelle 6.5, wobei die Default-MP aus Tabelle 6.2 simuliert wird. Der durchschnittliche Durchsatzwert ab $7200s$ beträgt über fünf Simulationen $D_m = 36.47\ 1/h$ und über 30 Simulationen $D_m = 36.45\ 1/h$. Der Unterschied liegt bei -0.05%. Der Erkenntnisgewinn wird damit als gering angesehen. Für die realitätsnahe Implementierung, bei der eine Simulation bis zu mehreren Stunden dauert, sind fünf Simulationen außerdem zeitgünstiger, was die Simulation von mehr unterschiedlichen Parameterkombinationen bzw. Produktionskonfigurationen in der vorliegenden Arbeit ermöglicht.

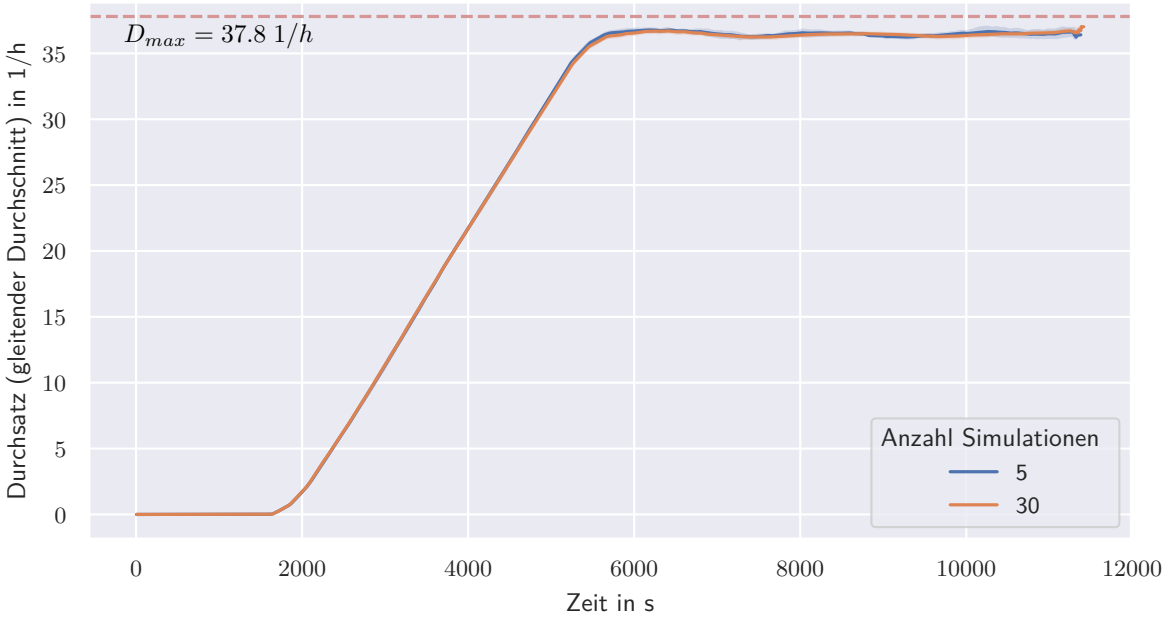


Abbildung C.2: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP aus Tabelle 6.2 mit 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, für $\Delta T_{\mathrm{GA}} = 200s$, fünf Simulationen vs. 30 Simulationen.

C.3 Dezentral: Auslegung der Vorausschau

Bei einer Analyse der Durchsatzdurchschnitte ab $7200s$ über jeweils fünf Simulationsläufe der Default-MP aus Tabelle 6.2 ergeben sich für den dezentralen Ansatz die Werte pro ΔT_{MAS} in Tabelle C.2.

Tabelle C.2: Durchschnittliche Durchsätze D_m im eingeschwungenen Zustand für die Simulationszeit ab $7200s$ in Abhängigkeit der Vorausschau ΔT_{MAS}, fünf Simulationsläufe pro ΔT_{MAS}.

Vorausschau ΔT_{MAS}	**Durchsatzdurchschnitt** D_m
$0s$	$27.78\ 1/h$
$10s$	$32.80\ 1/h$
$20s$	$36.10\ 1/h$
$30s$	$36.45\ 1/h$
$40s$	$36.26\ 1/h$
$50s$	$36.23\ 1/h$
$60s$	$36.30\ 1/h$
$70s$	$36.27\ 1/h$
$80s$	$36.10\ 1/h$
$90s$	$36.07\ 1/h$
$100s$	$35.93\ 1/h$

C.4 Dezentral: 30 im Vergleich zu fünf Simulationen

Abbildung C.3 vergleicht die Unterschiede von 30 zu fünf durchgeführten Simulationen im dezentralen Steuerungsansatz anhand der Default-Steuerung mit den Parametern aus Tabelle 6.6, wobei die Default-MP aus Tabelle 6.2 simuliert wird. Der durchschnittliche Durchsatzwert ab $7200s$ beträgt über fünf Simulationen $D_m = 36.45\ 1/h$ und über 30 Simulationen $D_m = 36.52\ 1/h$. Der Unterschied liegt bei 0.19%. Der Erkenntnisgewinn wird damit als gering angesehen. Für die realitätsnahe Implementierung, bei der eine Simulation bis zu mehreren Stunden dauert, sind fünf Simulationen außerdem zeitgünstiger, was die Simulation von mehr unterschiedlichen Parameterkombinationen bzw. Produktionskonfigurationen in der vorliegenden Arbeit ermöglicht.

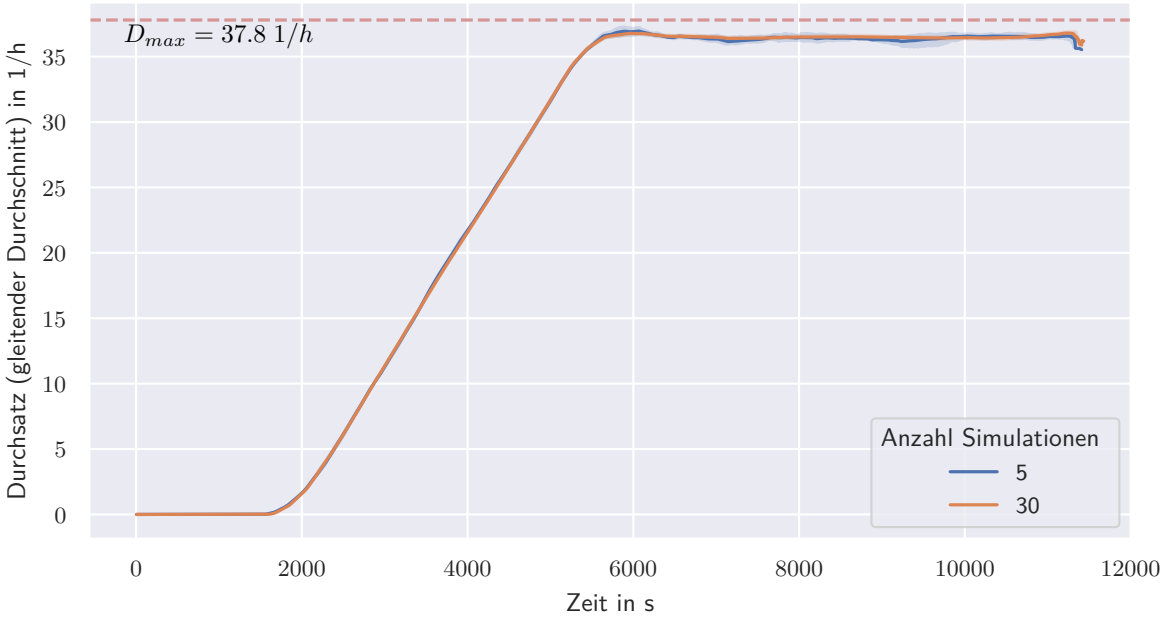


Abbildung C.3: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP aus Tabelle 6.2 mit 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, für $\Delta T_{\text{MAS}} = 30s$, fünf Simulationen vs. 30 Simulationen.

C.5 Lernend: PPO Lernkurve mit Action Masking

Abbildung C.4 dokumentiert die Entwicklung der Rewards pro Episode sowie die Entwicklung des Leistungsmaßes $L(\boldsymbol{\theta}', \boldsymbol{\theta})$ (vgl. (5.35)) pro Episode für die Lernphase des PPO-Agenten zur Steuerung der Default-MP aus Tabelle 6.2. Es gelten die Default-Parameter des Agenten aus Tabelle 6.7, wobei $n_{\text{episodes}} = 1000$ und $n_{\text{env}} = 2$.

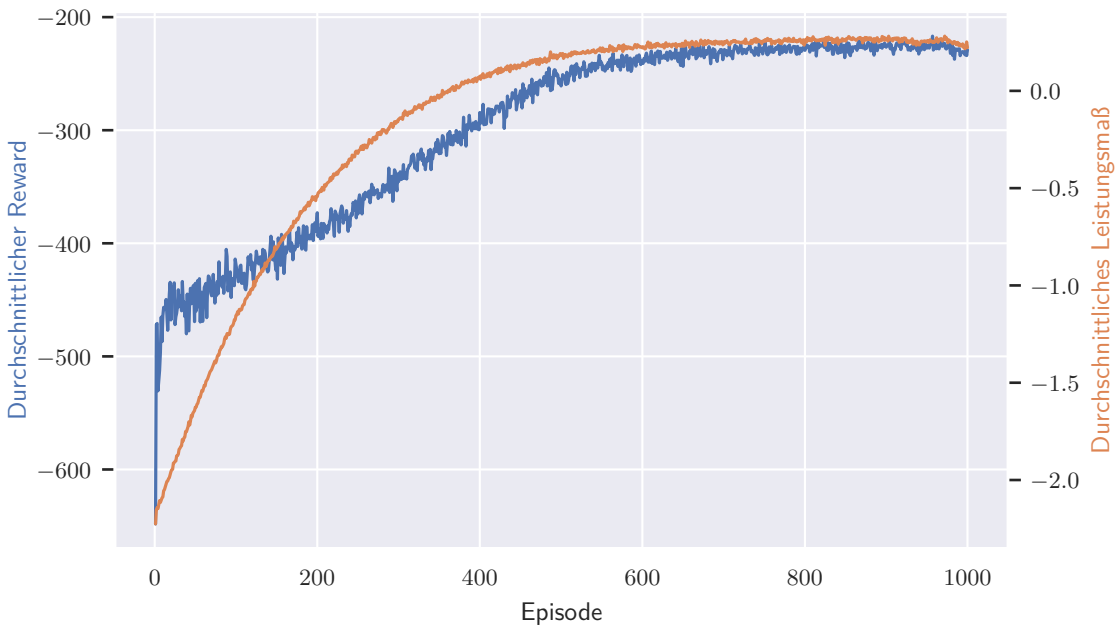


Abbildung C.4: Lernphase der Steuerung der Default-MP aus Tabelle 6.2 mit Action Masking für einen zufälligen Beispiel-Seed.

C.6 Lernend: PPO Lernkurve für unterschiedliche Anzahl von Umgebungen

Abbildung C.5 dokumentiert die durchschnittlichen Lernkurven der pro Umgebungszahl n_{env} durchgeführten fünf Lernläufe (fünf Seed-Einstellungen) der Default-MP aus Tabelle 6.2 für den PPO-Agenten. Es gelten die Default-Parameter des Agenten aus Tabelle 6.7, wobei $n_{\text{episodes}} = 1000$.

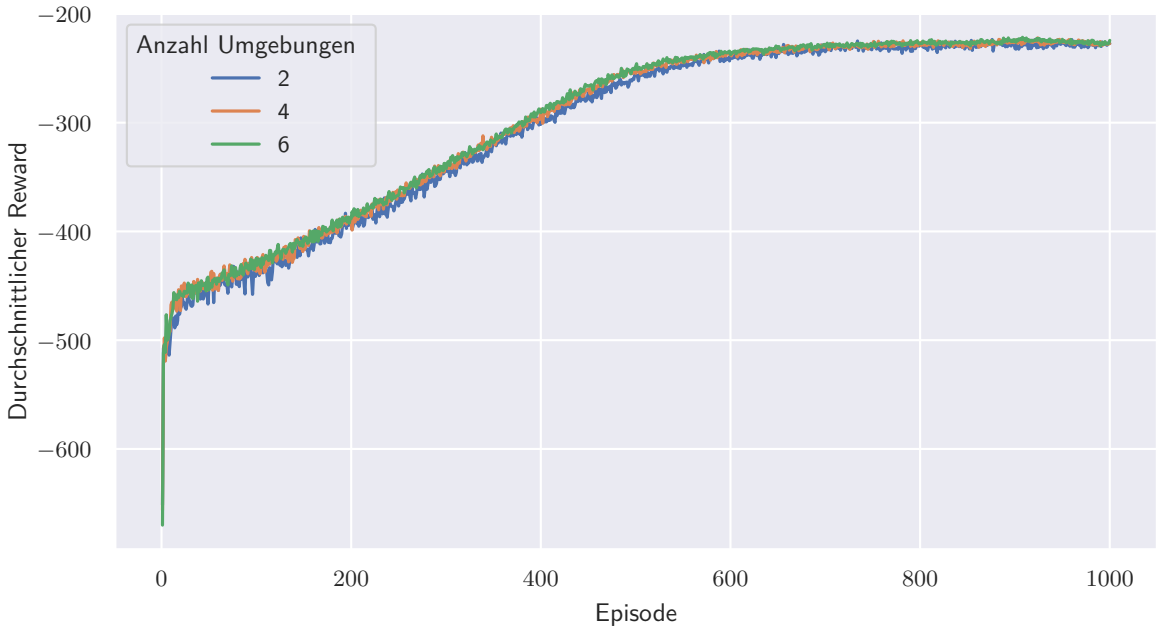


Abbildung C.5: Durchschnittliche Lernphase der Steuerung der Default-MP aus Tabelle 6.2 mit $n_{\text{env}} \in \{2, 4, 6\}$, fünf Seed-Einstellungen pro n_{env}.

C.7 Lernend: Durchsatzverläufe für unterschiedliche Anzahl von Umgebungen

Abbildung C.6 zeigt die Durchsatzergebnisse für die Simulation der Default-MP aus Tabelle 6.2 ohne Störungen in Abhängigkeit der Anzahl der Umgebungen nach einer Lernphase von $n_{\text{episodes}} = 1000$ Episoden mit dem PPO-Agenten. Für jedes n_{env} erfolgt die Lernphase mit fünf unterschiedlichen, vordefinierten Seeds auf Agentenseite. Die Abbildung stellt die gemittelten Simulationsergebnisse des gleitenden Durchsatzdurchschnitts über die Simulationszeit für die gelernten Policies π dar. Dabei unterscheiden sich die Ergebnisse für zwei, vier oder sechs parallele Umgebungen unwesentlich.

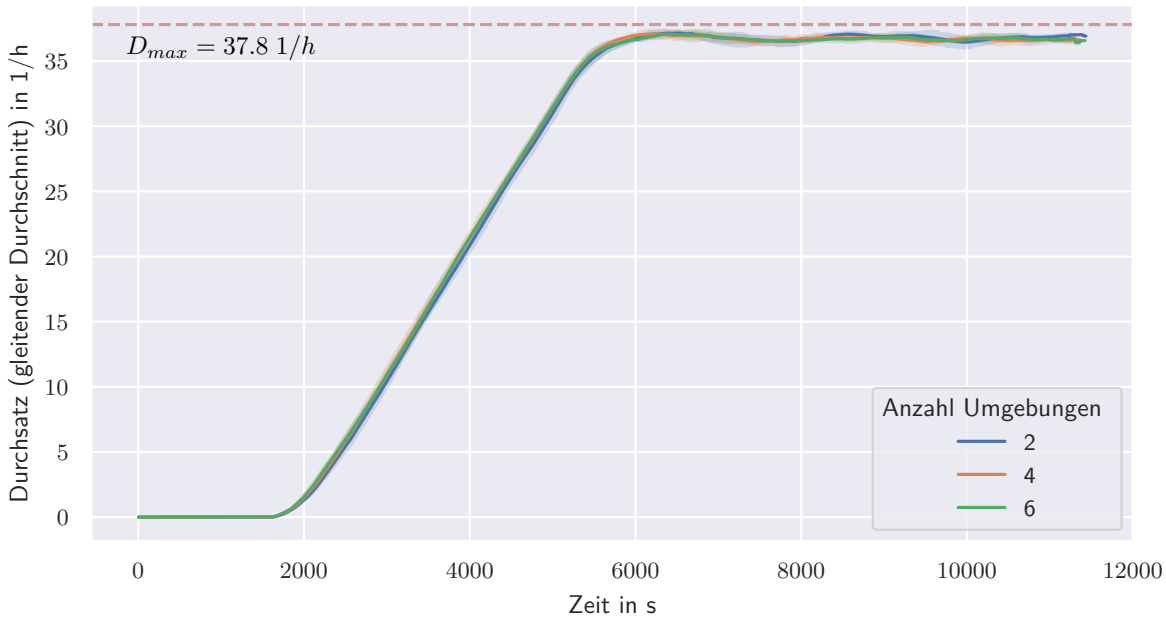


Abbildung C.6: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP aus Tabelle 6.2 mit 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, in Abhängigkeit der Anzahl der Umgebungen n_{env}, fünf unterschiedliche Seed-Einstellungen auf Agentenseite pro n_{env}.

C.8 Lernend: 30 im Vergleich zu fünf Simulationen

Abbildung C.7 vergleicht die Unterschiede von 30 zu fünf durchgeführten Simulationen im lernenden Steuerungsansatz anhand der Default-Steuerung mit den Parametern aus Tabelle 6.7 bei $n_{\text{env}} = 2$ und $n_{\text{episodes}} = 1000$, wobei die Default-MP aus Tabelle 6.2 simuliert wird. Jede Simulation repräsentiert eine eigene Seed-Einstellung in der Lernphase des Agenten. Der durchschnittliche Durchsatzwert ab $7200s$ beträgt über fünf Simulationen $D_m = 36.79\ 1/h$ und über 30 Simulationen $D_m = 36.64\ 1/h$. Der Unterschied liegt bei -0.4%. Der Erkenntnisgewinn wird damit als gering angesehen. Für die jeweils mehrere Stunden dauernde Lernphase sind fünf Simulationen außerdem zeitgünstiger, was die Simulation von mehr unterschiedlichen Parameterkombinationen bzw. Produktionskonfigurationen in der vorliegenden Arbeit ermöglicht.

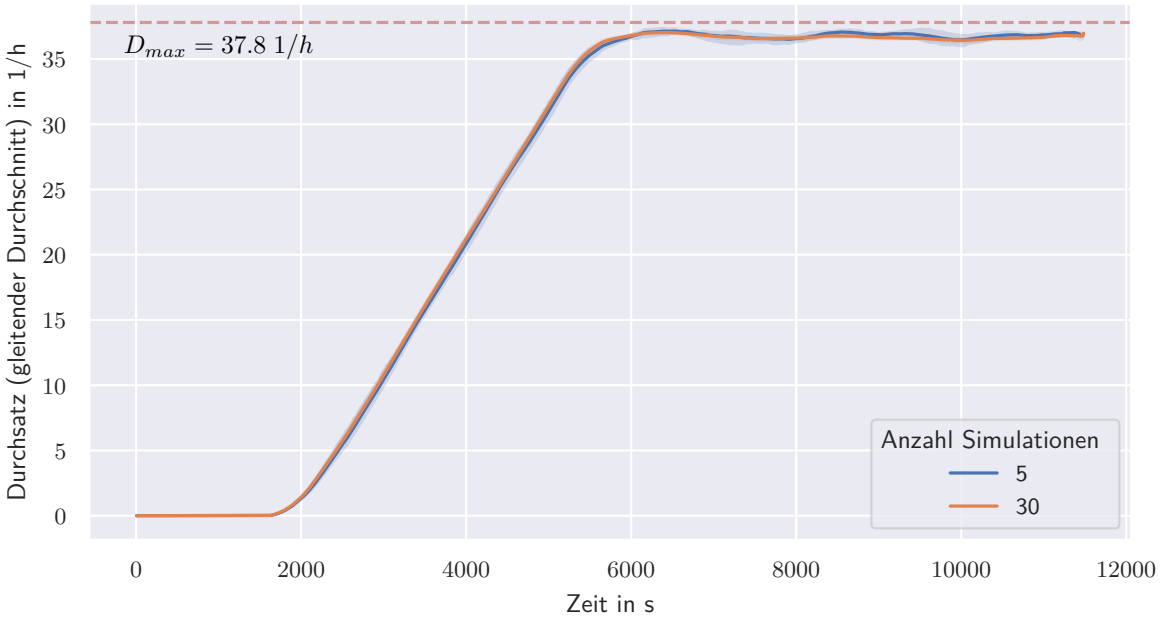


Abbildung C.7: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP aus Tabelle 6.2 mit 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, bei $n_{\text{env}} = 2$ und $n_{\text{episodes}} = 1000$, fünf Simulationen vs. 30 Simulationen.

D Vergleichende Experimente

D.1 Durchsatzverläufe im statischen Fall

Abbildung D.1 stellt die durchschnittlichen Durchsatzverläufe (jeweils fünf Simulationsläufe) für die Steuerung der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$ und $v_V = 2\ m/s$ dar, ohne dass Stationsstörungen auftreten.

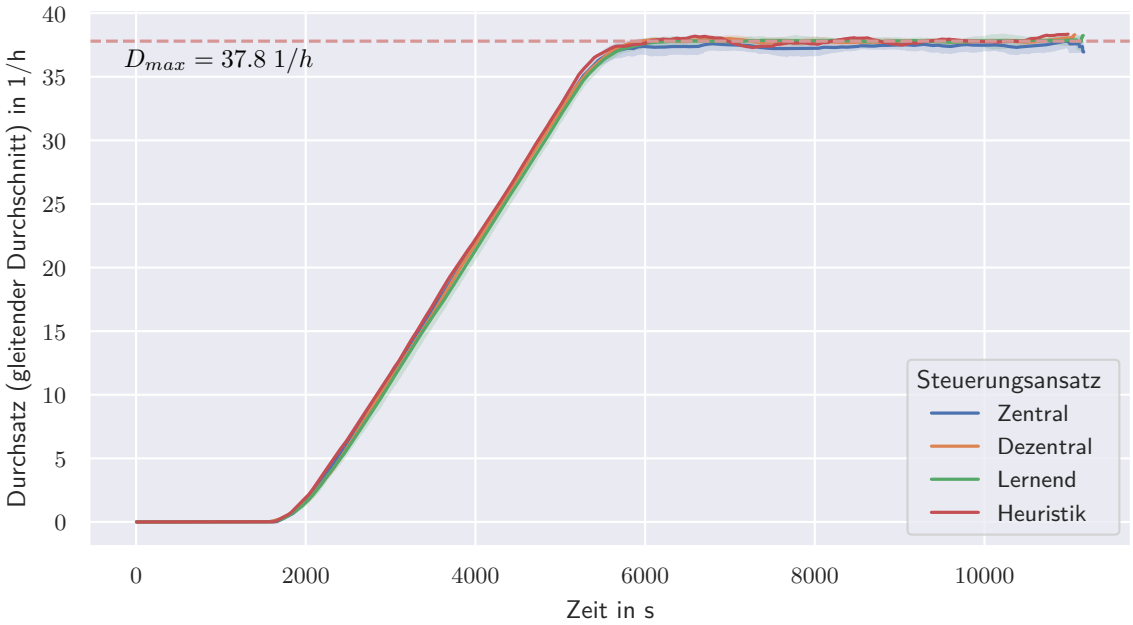


Abbildung D.1: Gleitender Durchsatzdurchschnitt für Simulation von 50 Mal Produkt PT_1 und 50 Mal Produkt PT_2 im 1:1-Produktmix, ohne Stationsstörungen, für Default-MP mit $n_V = n_{\mathrm{WT}} = 19$ und $v_V = 2\ m/s$, fünf Simulationsläufe pro Ansatz.

D.2 Simulations-Varianz durch unterschiedliche Störungs-Seeds, Verfügbarkeit 0.98

Abbildung D.2 zeigt die durchschnittlichen Durchsatzverläufe (jeweils fünf Simulationsläufe) für die Steuerung der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, für den zentralen Steuerungsansatz in Abhängigkeit der Störungs-Seeds.

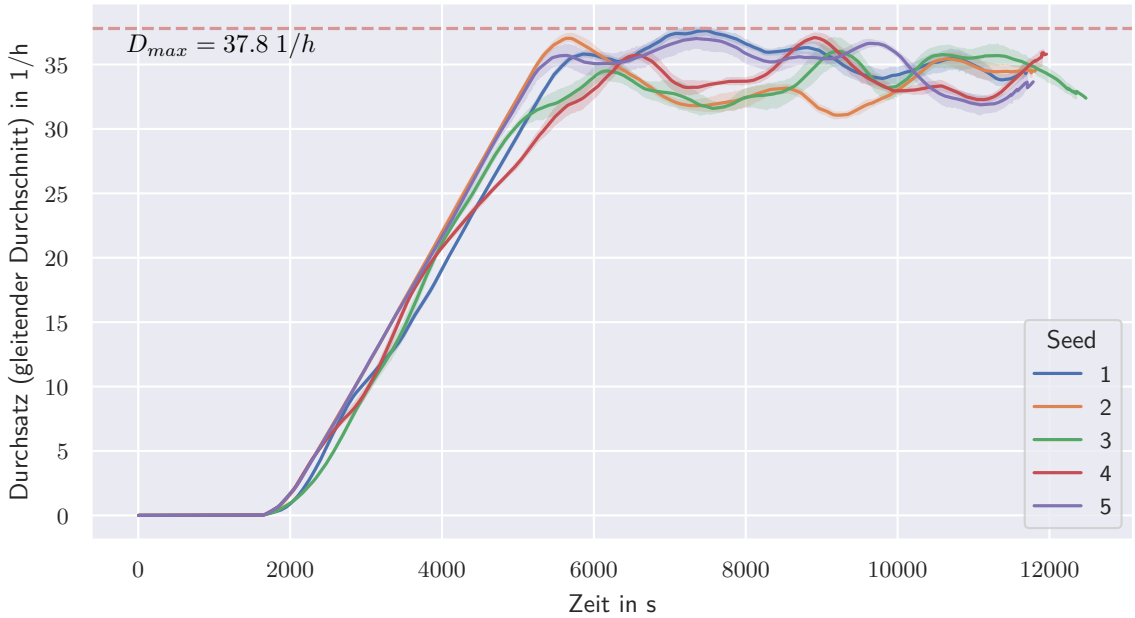


Abbildung D.2: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, für den zentralen Steuerungsansatz, fünf Simulationsläufe pro Störungs-Seed, in Abhängigkeit der Störungs-Seeds.

D.3 Durchlaufzeiten im dynamischen Fall, Verfügbarkeit 0.98

Abbildung D.3 zeigt die Verteilungen der Durchlaufzeiten über fünf Simulationsläufe pro Steuerungsansatz für die Steuerung der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed vier.

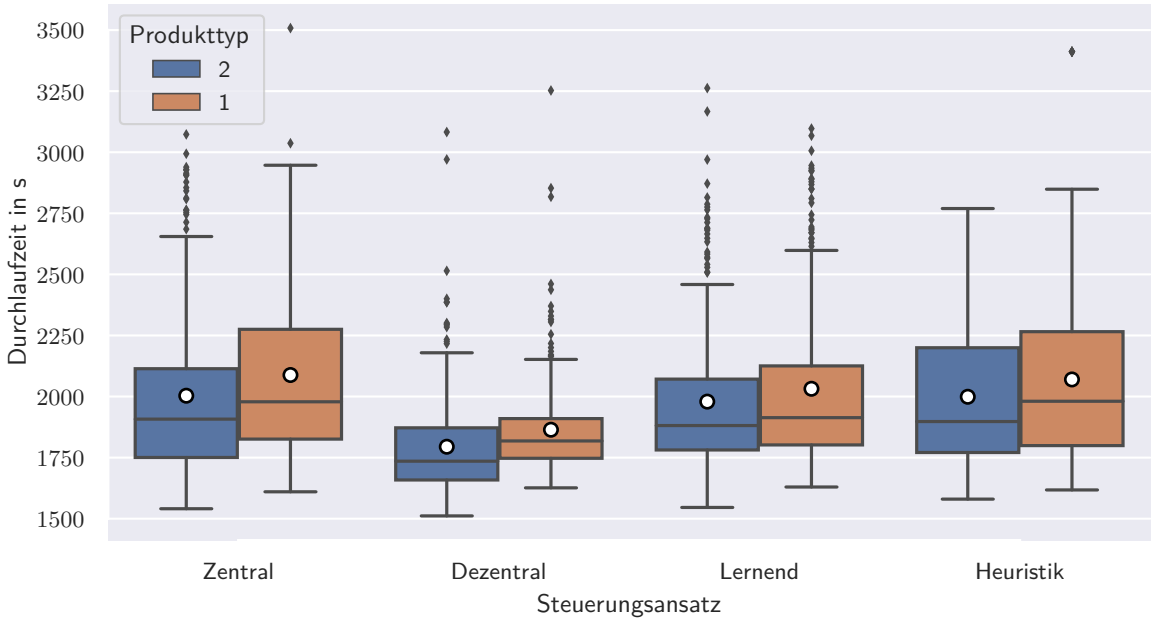


Abbildung D.3: Durchlaufzeitverteilungen aller Steuerungsansätze über alle fünf Simulationsläufe der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed vier.

D.4 Simulations-Varianz durch unterschiedliche Störungs-Seeds, Verfügbarkeit 0.90

Abbildung D.4 zeigt die durchschnittlichen Durchsatzverläufe (jeweils fünf Simulationsläufe) für die Steuerung der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.90$ und $MTTR = 600s$, $TTR \in [0, 900s]$, für den dezentralen Steuerungsansatz in Abhängigkeit der Störungs-Seeds.

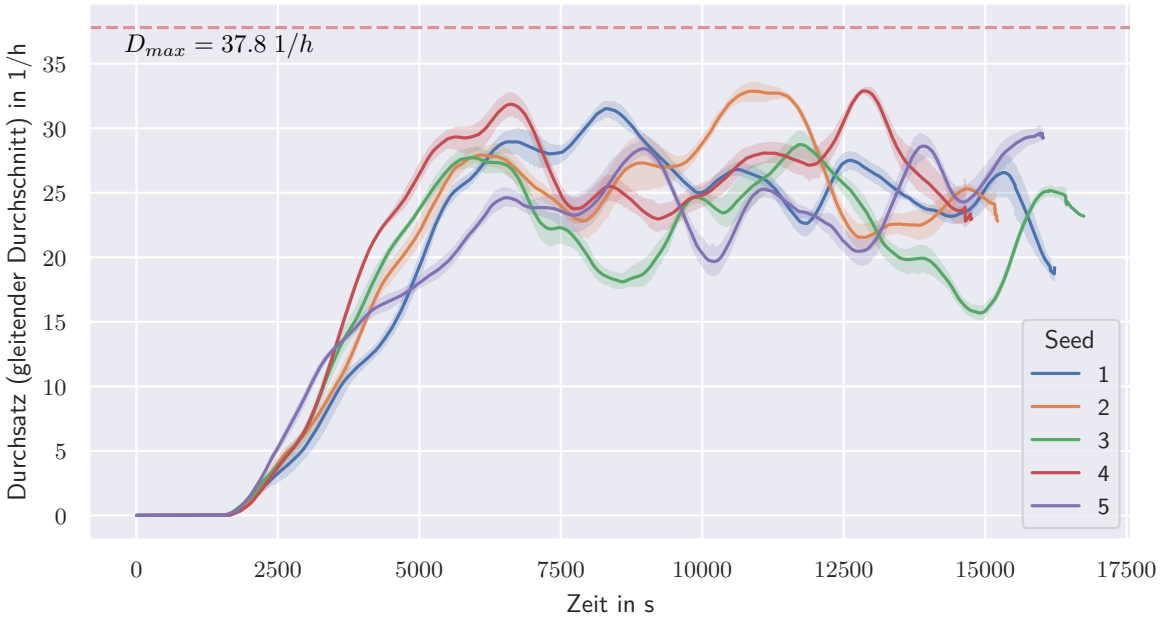


Abbildung D.4: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.90$ und $MTTR = 600s$, $TTR \in [0, 900s]$, für den dezentralen Steuerungsansatz, fünf Simulationsläufe pro Störungs-Seed, in Abhängigkeit der Störungs-Seeds.

D.5 Durchsatzverläufe im dynamischen Fall, Verfügbarkeit 0.90

Abbildung D.5 stellt die durchschnittlichen Durchsatzverläufe (jeweils fünf Simulationsläufe) für die Steuerung der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.90$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins dar.

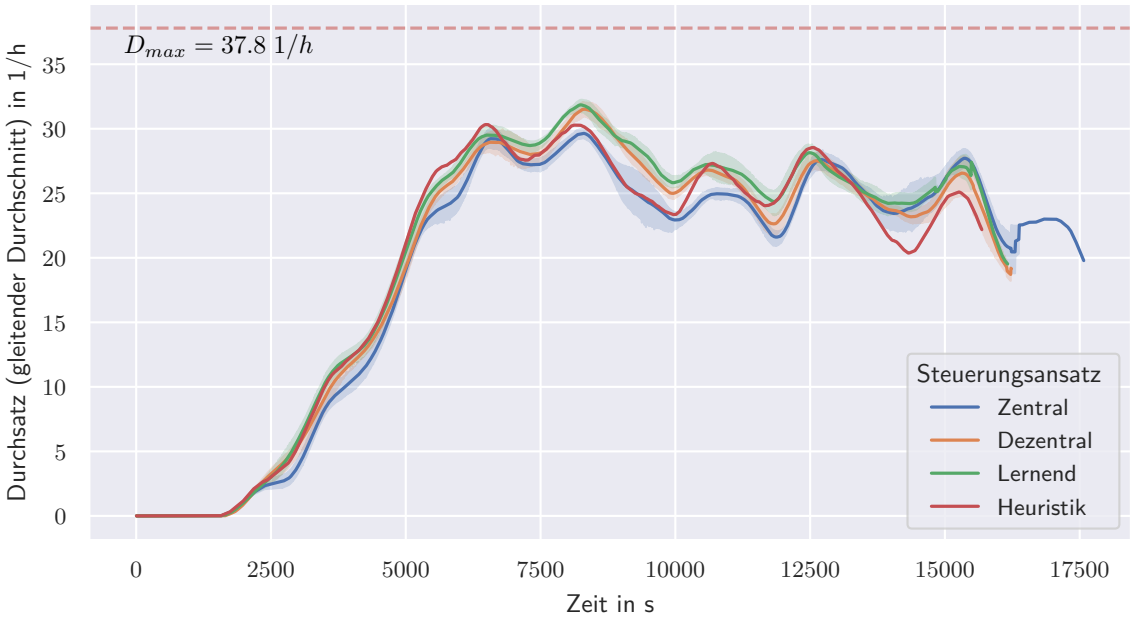


Abbildung D.5: Gleitender Durchsatzdurchschnitt für Simulation der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.90$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins, fünf Simulationsläufe pro Ansatz.

D.6 Lernend: Generalisierung der gelernten Modelle

Tabelle D.1 dokumentiert die Durchsatzergebnisse für die fünf Störungs-Seeds in der ursprünglichen und generalisierten Form des PPO-Agenten. Die ursprüngliche Form stellt die Ergebnisse für den Fall dar, dass der Agent bei dem jeweiligen Störungs-Seed angelernt und dann auch simulativ evaluiert wird. In der generalisierten Variante werden die fünf gelernten Modelle bei Störungs-Seed eins auf die anderen Störungs-Seeds angewendet. Deshalb gleichen sich die Ergebnisse der ersten Zeile bei Störungs-Seed eins. Die Durchsatzergebnisse entsprechen dem Durchschnitt des eingeschwungenen Zustands ab $7200s$ der Simulation. Die dokumentierten Durchsätze zeigen eindeutig die Fähigkeit des lernenden Ansatzes auf, das Gelernte auf unbekannte Daten anzuwenden. Sie zeigen seine Generalisierungsfähigkeit. Bei Störungs-Seed vier erzielt die generalisierte Variante sogar ein wesentlich besseres Durchsatzergebnis.

Tabelle D.1: Dynamische Simulationsergebnisse der lernenden Steuerung in der Default-MP mit $n_V = n_{\mathrm{WT}} = 19$, $V_M = 0.90$ und $MTTR = 600s$, $TTR \in [0, 900s]$, für fünf unterschiedliche Störungs-Seeds, wobei im generalisierten Fall die bei Störungs-Seed eins angelernten PPO-Modelle auf die Störungs-Seeds zwei, drei, vier und fünf Anwendung finden.

		Ursprünglich	**Generalisiert**
Nr.	Seed	D_m $[1/h]$	D_m $[1/h]$
1	1	26.93	26.93
2	2	26.66	26.75
3	3	22.88	22.79
4	4	27.39	28.01
5	5	26.21	26.10

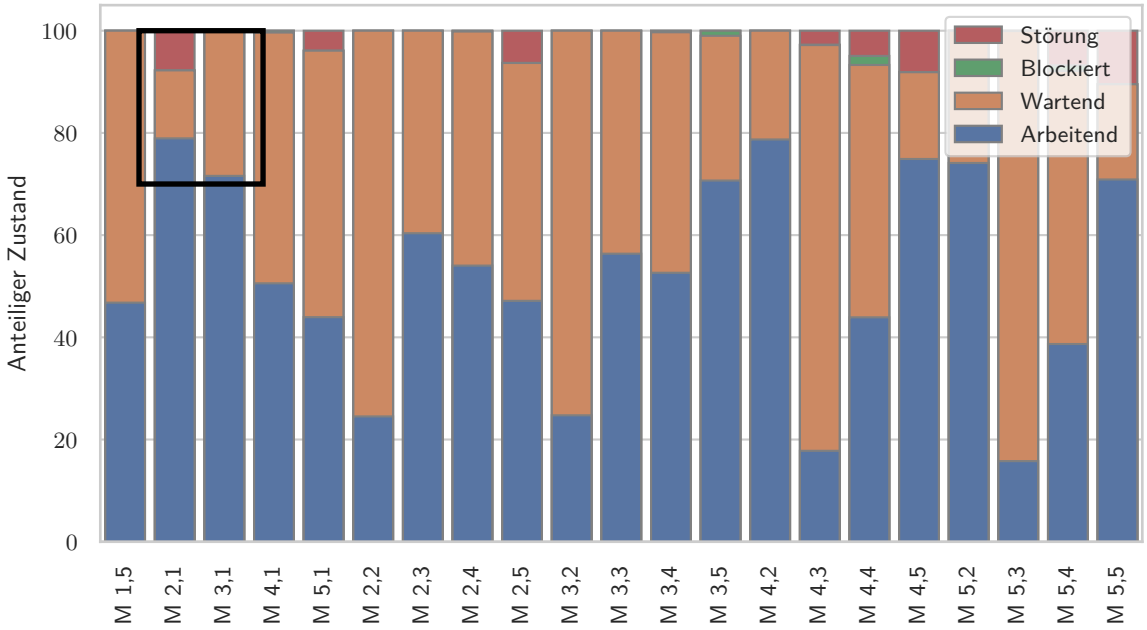
E Generalisierende Experimente

E.1 Beispiel 2: Stationsauslastungen, Verfügbarkeit 0.98

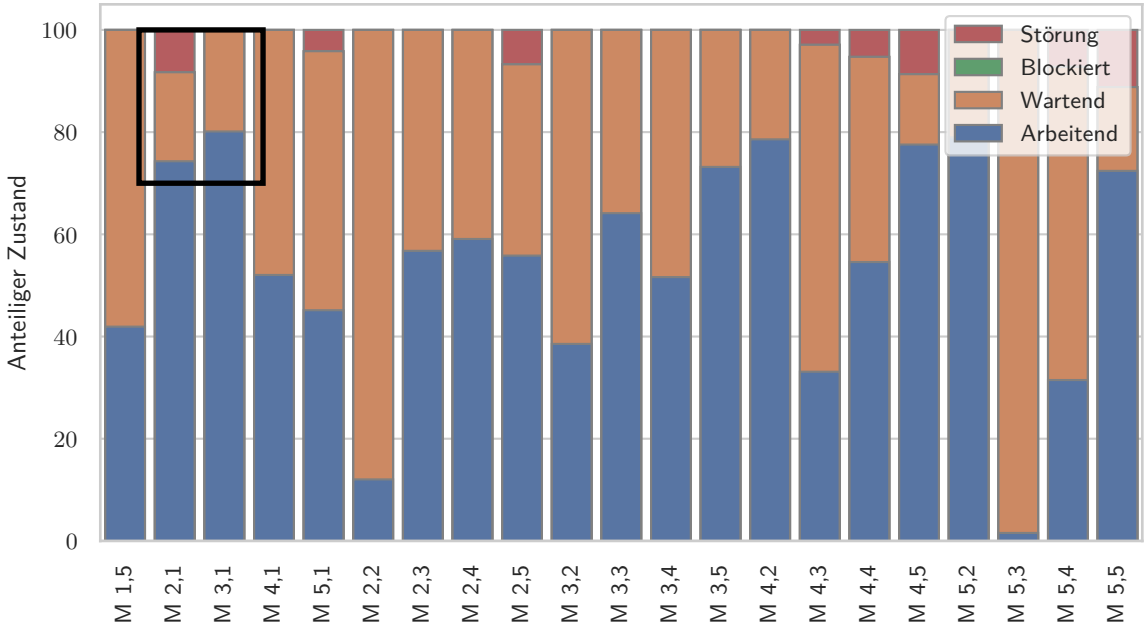
Abbildung E.1a zeigt die Auslastungen für die dynamischen Simulationen in Evaluationsbeispiel 2 mit dezentraler Steuerung. Abbildung E.1b stellt die Auslastungen der lernenden Steuerung gegenüber[1]. Die Darstellung dokumentiert für Störungs-Seed eins die durchschnittlichen Auslastungen ab einer Simulationszeit von $7200s$, weshalb auch die Endphase der Produktion in die Auslastungswerte einfließt. In der Endphase findet jedoch nach und nach die Fertigstellung der letzten Aufträge statt. Dadurch reduzieren sich die Stationsauslastungen. Der relative Vergleich zwischen dezentraler und lernender Steuerung verdeutlicht trotzdem die Zusammenhänge aus Abbildung 6.16.

Bei der Simulation mit Störungs-Seed eins unterliegt Station $M_{2,1}$ einer bzw. mehreren Störungen. $M_{3,1}$ bleibt hingegen für diesen Simulationslauf störungsfrei, wie die Abbildungen zeigen. Der dezentrale Ansatz bevorzugt trotz der Störung die Kapazität der teilweise gestörten Station $M_{2,1}$, was die Verläufe im Sankey-Diagramm (vgl. Abbildung 6.16) widerspiegelt. Der lernende Ansatz sorgt hingegen dafür, dass $M_{3,1}$ stärker genutzt wird als $M_{2,1}$. Die Darstellung lässt vermuten, dass der PPO-Agent während der Störungszeit bestmöglich auf Station $M_{3,1}$ ausweicht. Außerdem ist die Transportzeit von der Quelle zu $M_{3,1}$ geringer als zu $M_{2,1}$. Damit ist es je nach Störungslage grundsätzlich sinnvoll, relativ gesehen mehr Kapazität an $M_{3,1}$ zu nutzen.

[1] In beiden Darstellungen sind die Stationen $M_{1,1}$, $M_{1,2}$, $M_{1,3}$ und $M_{1,4}$ ausgeblendet, weil sie im MP-Szenario aus Tabelle 6.1 nicht im Einsatz sind.



(a) **Dezentral**



(b) **Lernend**

Abbildung E.1: Darstellung der Auslastungen für das MP-Szenario mit drei Produkttypen (Evaluationsbeispiel 2), $n_V = n_{\mathrm{WT}} = 19$ Fahrzeugen und WTs, im dynamischen Fall mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins, ab $7200s$ Simulationszeit, über fünf Simulationen pro Ansatz.

E.2 Beispiel 2: Durchsatzverläufe und Durchlaufzeiten, Verfügbarkeit 0.98

Abbildung E.2 zeigt die Verteilungen der Durchlaufzeiten über fünf Simulationsläufe pro Steuerungsansatz für die Steuerung von Evaluationsbeispiel 2 mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins. Abbildung E.3 stellt die zugehörigen Durchsatzverläufe dar.

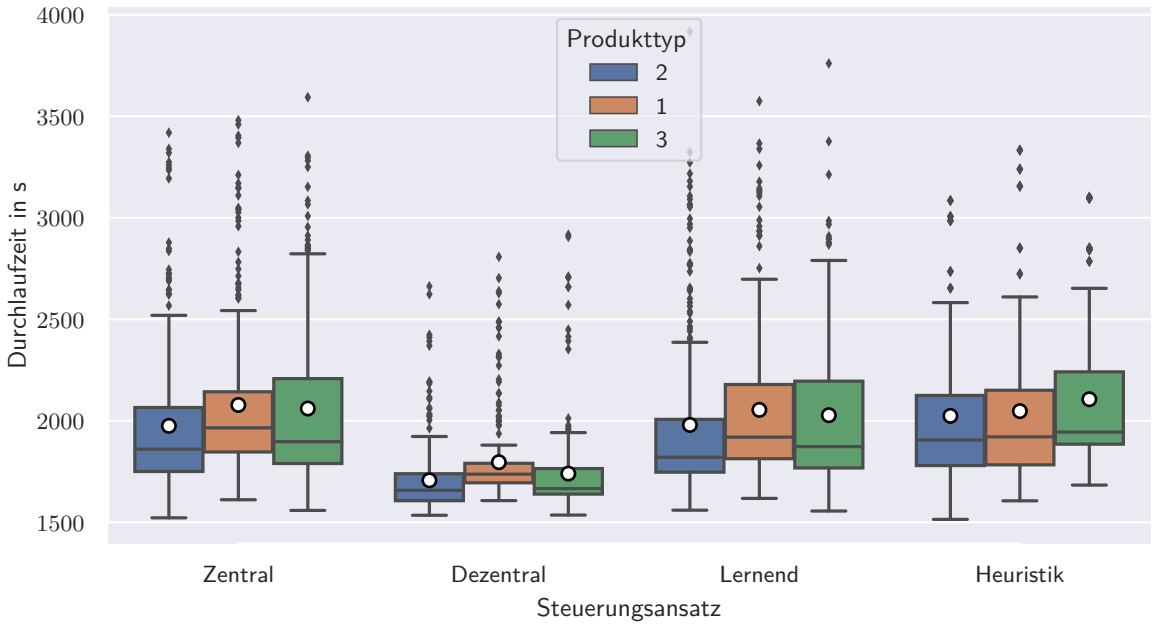


Abbildung E.2: Durchlaufzeitverteilungen aller Steuerungsansätze über alle fünf Simulationsläufe der MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins, generalisiertes Evaluationsbeispiel 2.

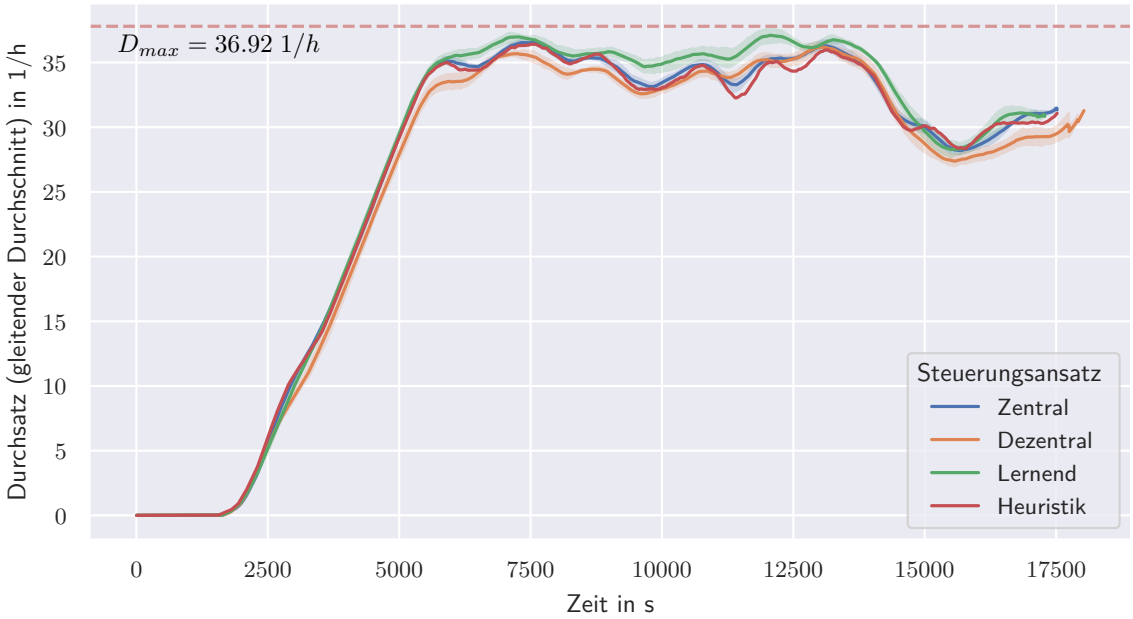


Abbildung E.3: Gleitender Durchsatzdurchschnitt für Simulation der MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins, fünf Simulationsläufe pro Ansatz, generalisiertes Evaluationsbeispiel 2.

E.3 Beispiel 3: Durchsatzverläufe und Durchlaufzeiten, Verfügbarkeit 0.98

Abbildung E.4 zeigt die Verteilungen der Durchlaufzeiten über fünf Simulationsläufe pro Steuerungsansatz für die Steuerung von Evaluationsbeispiel 3 mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins. Abbildung E.5 stellt die zugehörigen Durchsatzverläufe dar.

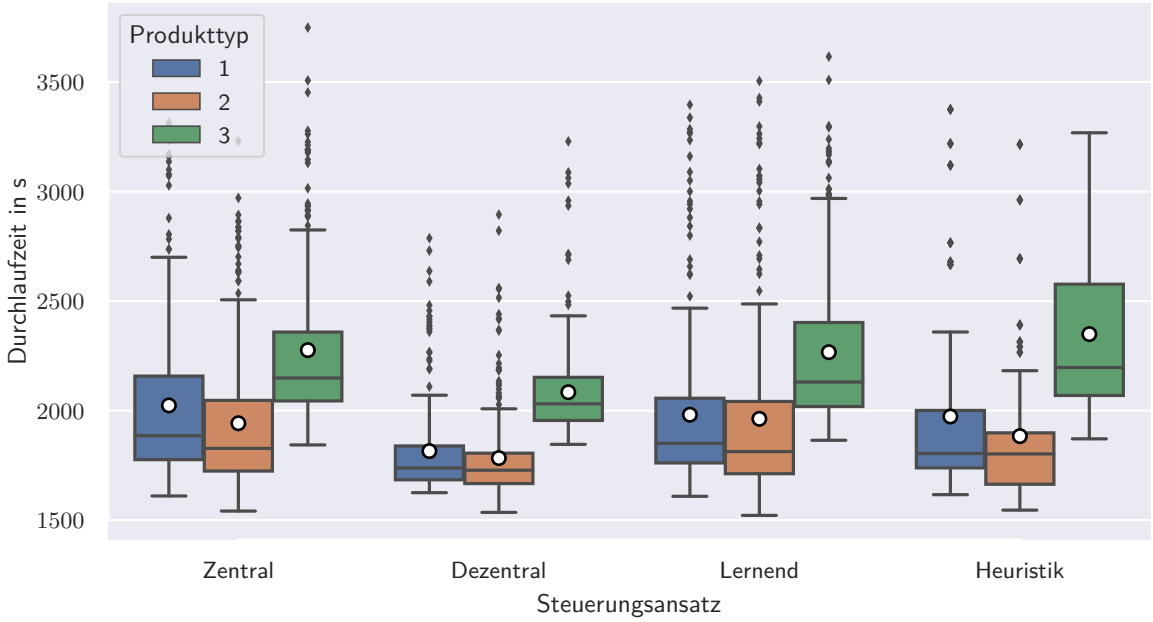


Abbildung E.4: Durchlaufzeitverteilungen aller Steuerungsansätze über alle fünf Simulationsläufe der MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins, generalisiertes Evaluationsbeispiel 3.

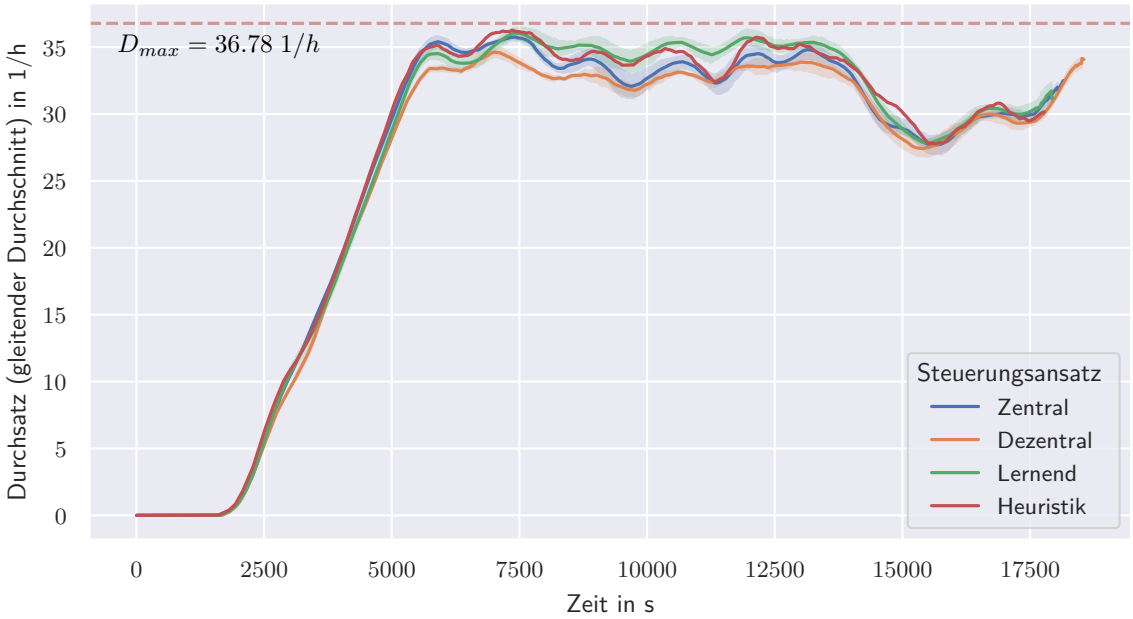


Abbildung E.5: Gleitender Durchsatzdurchschnitt für Simulation der MP mit $n_V = n_{\mathrm{WT}} = 19$, Stationsstörungen mit $V_M = 0.98$ und $MTTR = 600s$, $TTR \in [0, 900s]$, bei Störungs-Seed eins, fünf Simulationsläufe pro Ansatz, generalisiertes Evaluationsbeispiel 3.

Abkürzungen

A-HEFT	Adaptive Heterogeneous Earliest Finish Time
AC-PPO	Actor-Critic Proximal Policy Optimization
ADACOR	ADAptive holonic COntrol aRchitecture
AE-Agent	Auftragseinlastungs-Agent
AP	Ausgangspuffer
ARENA	Active Research Environment for the Next Generation of Automobiles
BDE	Betriebsdatenerfassung
BMBF	Bundesministerium für Bildung und Forschung
BOA	Belastungsorientierte Auftragsfreigabe
CC	Cloud Computing
CNC	Computerized Numerical Control
CNP	Contract Net Protocol
Conwip	Constant Work in Process
CPPS	Cyber-physisches Produktionssystem
CPS	Cyber-physisches System
DES	Discrete-event simulation
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
E-Achse	Elektrische Achse
EP	Eingangspuffer
ERP	Enterprise Resource Planning
FCFS	First Come First Served
FIFO	First-In-First-Out
FIPA	Foundation for Intelligent Physical Agents
FMS	Flexible manufacturing system
FTF	Fahrerloses Transportfahrzeug
FTS	Fahrerloses Transportsystem
GA	Genetischer Algorithmus
GAE	Generalized Advantage Estimation

HIL	Hardware-in-the-Loop
IB	In Bearbeitung
IdD	Internet der Dinge
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
M-Agent	Maschinen-Agent
MAS	Multiagentensystem
MDP	Markov Decision Process
MES	Manufacturing Execution System
MIL	Model-in-the-Loop
MLP	Multilayer Perceptron
MP	Modulare Produktion
MQA	Mittlere quadratische Abweichung
MQTT	Message Queuing Telemetry Transport
MTBF	Mean Time Between Failure
MTTR	Mean Time To Restoration
NN	Neuronales Netz
NP	Nicht-deterministische Polynomialzeit
Op-H	Operational holon
OPC UA	Open Platform Communications Unified Architecture
Ord-H	Order holon
P-Agent	Produkt-Agent
PPO	Proximal Policy Optimization
PPS	Produktionsplanung und -steuerung
Prod-H	Product holon
PROSA	Product-Resource-Order-Staff Architecture
ReLU	Rectified Linear Units
Res-H	Resource holon
RL	Reinforcement Learning
SCADA	Supervisory Control and Data Acquisition
Serva TS	Serva Transport Systems
SIL	Software-in-the-Loop
SMART	Semi-Markov Average Reward Technique
SPS	Speicherprogrammierbare Steuerung
SQ	Shortest Queue
Staff-H	Staff holon

Sup-H	Supervisor holon
Task-H	Task holon
TCP/IP	Transmission Control Protocol/Internet Protocol
TD	Temporal Difference
TRPO	Trust Region Policy Optimization
TTR	Time To Restoration
V-Agent	Fahrzeug-Agent
VIBN	Virtuelle Inbetriebnahme
VM	Virtuelle Maschine
WT	Werkstückträger

Symbole

$\mathcal{A}$	Menge aller möglichen Actions
a	Action aus der Menge $\mathcal{A}$
α	Lernrate
A_m	Mittlere Auslastung
A_{M_k}	Action zu Station M_k
A_{Quelle_x}	Action zu Quelle für Produkttyp PT_x
A_{Senke}	Action zu Senke
a_t	Action-Sample in der Umgebung
$A_t^{\text{GAE}(\gamma,\lambda)}$	Advantage-Estimate in Zeitschritt t
$A_{t,\text{val}}$	Menge valider Actions in Zeitschritt t
$\boldsymbol{b}$	Vektor der eingehenden Gebote b_i
β	Gewichtungsfaktor der GA-Fitnessfunktion
b_i	Gebot des Agenten i
c_1	Gewichtung des Value-Funktions-Terms
c_2	Gewichtung des Entropieterms
C_i	Produktionsspanne eines Auftrags J_i
c_i	Leerlaufkosten einer Station aufgrund des Agenten i
C_{max}	Produktionsspanne aller Aufträge
D	Definitionsmenge
δ	Gewichtungsfaktor im Gebot b_i
δ_t	Temporal Difference-Fehler in Zeitschritt t
ΔT_{AE}	Mindestabstand zwischen zwei Einlastungen
ΔT_{GA}	Vorausschau bei der Schedule-Berechnung
$\Delta T_{\text{GA},min}$	Mindest-Vorausschau bei der Schedule-Berechnung
ΔT_{MAS}	Vorausschau einer Station für Auktionen
Δt_s	Abstand zwischen zwei Schedule-Berechnungen
DFZ_{ipl}	Durchführungszeit eines Vorgangs O_{ipl}
D_m	Mittlerer Produktionsdurchsatz
D_{max}	Maximaler Produktionsdurchsatz
DZ_{ipl}	Vorgangsdurchlaufzeit einer Operation O_{ipl}
DZ_{J_i}	Durchlaufzeit eines Auftrags J_i

$DZ_{J,m}$	Mittlere Durchlaufzeit aller Aufträge in J
$DZ_{J,m,min}$	Minimal erreichbare mittlere Durchlaufzeit aller Aufträge in J
E	Menge an Kanten e
e	Kante aus der Menge E
E_{ipl}	Bearbeitungsende eines Vorgangs O_{ipl}
E_{J_i}	Fertigstellungszeitpunkt eines Auftrags J_i
ϵ	Clip-Parameter
F	Menge an Fähigkeiten
F_{ipl}	Menge benötigter Fähigkeiten für Operation O_{ipl}
F_{M_k}	Menge an Fähigkeiten von Station M_k
γ	Discount-Faktor
G_t	Return
h_{in}	Anzahl an Neuronen der Input-Schicht
h_{out}	Anzahl an Neuronen der Output-Schicht
I	Individuum
i	Einem Auftrag J_i zugeordneter Agent
I_{max}	Individuum mit bestem Fitnesswert
$Items(I)$	Menge an Schedule-Items des Individuums I
J	Menge an Aufträgen J_i
j	Anzahl antwortender Auktionsteilnehmer
J_{akt}	Menge aktiver Produktionsaufträge J_i
$J(I)$	Fitness eines Individuums I
J_i	Produktionsauftrag aus der Menge J
J_{max}	Maximaler Fitnesswert
$J_{max,m}$	Mittelwert der erreichten maximalen Fitnesswerte
$J(\boldsymbol{\theta})$	Leistungsmaß der Policy in Abhängigkeit des Parametervektors $\boldsymbol{\theta}$
$J(\boldsymbol{\theta}', \boldsymbol{\theta})$	Leistungsmaß der Policy in Abhängigkeit der Parametervektoren $\boldsymbol{\theta}'$ und $\boldsymbol{\theta}$
K	Kindindividuum
k	Anzahl abgelehnter Auktionsteilnehmer
l	Anzahl akzeptierter Auktionsteilnehmer
λ	Discount-Faktor (GAE-Parameter)
λ_0	Initiale Population
λ_i	Population der i-ten Generation
$\lambda_{J,m}$	Mittlere Ankunftsrate der Aufträge J
$L^{clip}_{sur,\tau}(\boldsymbol{\theta}', \boldsymbol{\theta})$	Clipped-Surrogate-Objective für Trajektorie τ in Abhängigkeit der Parametervektoren $\boldsymbol{\theta}'$ und $\boldsymbol{\theta}$

$L_{sur}^{clip}(\boldsymbol{\theta}', \boldsymbol{\theta})$	Clipped-Surrogate-Objective in Abhängigkeit der Parametervektoren $\boldsymbol{\theta}'$ und $\boldsymbol{\theta}$
$L_{sur}(\boldsymbol{\theta}', \boldsymbol{\theta})$	Surrogate-Objective in Abhängigkeit der Parametervektoren $\boldsymbol{\theta}'$ und $\boldsymbol{\theta}$
$L(\boldsymbol{\theta}', \boldsymbol{\theta})$	Gesamtes Leistungsmaß in Abhängigkeit der Parametervektoren $\boldsymbol{\theta}'$ und $\boldsymbol{\theta}$
$L^{\mathrm{VF}}(\boldsymbol{\theta}', \boldsymbol{\theta})$	Leistung der Value-Funktion in Abhängigkeit der Parametervektoren $\boldsymbol{\theta}'$ und $\boldsymbol{\theta}$
M	Menge modularer Stationen M_k
m	Anzahl potenzieller Auktionsteilnehmer
M_{ipl}	Menge an Stationen mit Fähigkeiten F_{ipl}
m_{ipl}	Anzahl an Stationen mit Fähigkeiten F_{ipl}
M_k	Station aus der Menge M
$M_{k,ipl}$	Gewählte Station M_k mit Fähigkeiten F_{ipl}
$m_{k,\mathrm{stoer}}$	State-Information zu Störung an Station M_k
$m_{k,\mathrm{WZ}}$	State-Information zu Wartezeit an Station M_k
$M_{m,n}$	Station mit Rasterkoordinaten m und n
$MTBF$	Mean Time Between Failure
$MTTR$	Mean Time To Restoration
$\mu(s)$	Eintrittswahrscheinlichkeit eines States s
n	Anzahl bietender Auktionsteilnehmer
n_a	Anzahl verfügbarer Actions in $\mathcal{A}$
N_{AB}	Anzahl der Abgänge im Bezugszeitraum
n_{AP}	Kapazität des Ausgangspuffers
n_{env}	Anzahl paralleler Umgebungen
n_{EP}	Kapazität des Eingangspuffers
n_{episodes}	Anzahl an durchlaufenen Episoden
n_{epochs}	Anzahl an Updates pro Lernvorgang
N_{GEN}	Anzahl durchlaufener Generationen
n_J	Anzahl an Produktionsaufträgen in J
n_{J,M_k}	Anzahl eingeplanter Aufträge J an Station M_k
n_M	Anzahl modularer Stationen in M
n_O	Anzahl an Operationen in O_{ip}
n_P	Anzahl an Prozessplänen in P_i
N_{POP}	Populationsgröße
n_{PT}	Anzahl an Produkttypen in PT
n_R	Anzahl an Routen in R_i
n_s	Anzahl an State-Informationen in s_t
N_{SEL}	Anzahl selektierter Individuen

n_τ	Länge einer Trajektorie τ
n_V	Anzahl an Fahrerlosen Transportfahrzeugen in V
$n_{V,min}$	Minimal notwendige Fahrzeuge V zum Erreichen des Durchsatzoptimums
n_{WT}	Anzahl an Werkstückträgern in WT
n_{zuk}	Anzahl an zu sammelnden Zukunftserfahrungen für korrektes Lernen
O_{ip}	Menge an Operationen in P_{ip}
O_{ipl}	Operation aus der Menge O_{ip}
O_{ipn_O}	Letzte Operation aus der Menge O_{ip}
P	Elternindividuum
p	Preis des versteigerten Guts
PB	Produktionsbestand
PB_m	Mittlerer Produktionsbestand
PB_{max}	Maximaler Produktionsbestand
PB_{min}	Mindestens notwendiger Produktionsbestand zum Erreichen des Durchsatzoptimums
pf_l	State-Information zu Fortschritt l des Auftrags
P_i	Menge an Prozessplänen von Auftrag J_i
π	Policy
π_*	Optimale Policy
$\pi(a \mid s)$	Wahrscheinlichkeit in State s Action a zu wählen
$\pi(a \mid s, \boldsymbol{\theta})$	Wahrscheinlichkeit in State s mit dem Parametervektor $\boldsymbol{\theta}$ Action a zu wählen
P_{ip}	Prozessplan aus der Menge P_i
$\pi_{\boldsymbol{\theta}}$	Policy des Parametervektors $\boldsymbol{\theta}$
$\pi_{\boldsymbol{\theta}}(a \mid s)$	Wahrscheinlichkeit in State s mit dem Parametervektor $\boldsymbol{\theta}$ Action a zu wählen
p_M	Mutationswahrscheinlichkeit
p_R	Rekombinationswahrscheinlichkeit
Ψ_t	Allgemeine Funktion für Güte einer Action a_t
$p(s', r \mid s, a)$	Wahrscheinlichkeit aus State s mit Action a in State s' mit Reward r überzugehen
$p(s' \mid s, a)$	Wahrscheinlichkeit aus State s mit Action a in State s' überzugehen
PT	Menge an Produkttypen PT_x
$p(\tau \mid \pi_{\boldsymbol{\theta}})$	Eintrittswahrscheinlihckeit einer Trajektorie τ unter Policy $\pi_{\boldsymbol{\theta}}$
$p_{t:T-1}$	Wahrscheinlichkeitsverhältnis zweier Trajektorien

PT_x	Produkttyp aus der Menge PT
q	Anzahl ablehnender Auktionsteilnehmer
$q_\pi(s,a)$	Action-Value-Funktion (Wert) von State s mit Action a und anschließend Policy π folgend
$\mathcal{R}$	Menge aller möglichen Rewards
r	Reward aus der Menge $\mathcal{R}$
r_{bonus}	Reward zur Belohnung des Agenten
R_i	Menge an Routen von Auftrag J_i
R_{ib}	Route aus der Menge R_i
R_{i,t_s}	Menge an Routen von Auftrag J_i zum Zeitpunkt t_s
r_{punish}	Reward zur Bestrafung des Agenten
$r(s,a,s')$	Erwarteter Reward einer gegebenen Transition
r_t	Reward-Sample aus der Umgebung
$R_t(\boldsymbol{\theta}',\boldsymbol{\theta})$	Verhältnis der Eintrittswahrscheinlichkeiten einer Action a_t in Abhängigkeit der Parametervektoren $\boldsymbol{\theta}'$ und $\boldsymbol{\theta}$
$\mathcal{S}$	Menge aller möglichen States
s	State aus der Menge $\mathcal{S}$
S_{ipl}	Bearbeitungsstart eines Vorgangs O_{ipl}
S_{J_i}	Einlastungszeitpunkt eines Auftrags J_i
sq_x	State-Information zu Verfügbarkeit von Produkttyp PT_x in der Quelle
s_t	State-Sample der Umgebung
$S(\boldsymbol{\theta}')$	Entropie von Parametervektor $\boldsymbol{\theta}'$
T	Zeitschritt des Terminal-States
t	Zeitschritt der Umgebung
$t_{\text{AP},ipl}$	Liegezeit im Ausgangspuffer nach Operation $O_{ip(l-1)}$
$t_{\text{AP},min}$	Mindest-Transportzeit im Ausgangspuffer
$t_{\text{AP,Rest}}$	Verbleibende Transportzeit im Ausgangspuffer
τ	Trajektorie in der Umgebung
t_{avg}	Wert zur Normalisierung von Agenteninformationen
t_{BZ}	Länge des Bezugszeitraums
T_{dead}	Zeitliche Auktionsdeadline
$t_{e,\text{ende}}$	Frühestmögliche Endzeit der Kante e
$t_{\text{EP},ipl}$	Liegezeit im Eingangspuffer vor Operation O_{ipl}
$t_{\text{EP},min}$	Mindest-Transportzeit im Eingangspuffer
$t_{\text{EP,Rest}}$	Verbleibende Transportzeit im Eingangspuffer
$t_{e,\text{start}}$	Frühestmögliche Startzeit der Kante e
t_{frei}	Zeitpunkt freier Kapazität einer Station

$\boldsymbol{\theta}$	Parametervektor der Policy
$\boldsymbol{\theta}_t$	Parametervektor der Policy zum Zeitpunkt t
T_i	Mathematischer Wurzelbaum eines Auftrags J_i
T_i'	Teilgraph des Wurzelbaums T_i eines Auftrags J_i
$t_{i,\text{ank}}$	Geschätzte Ankunftszeit eines Agenten i
t_{ipl}	Bearbeitungszeit von Operation O_{ipl}
$t_{ipl,k}$	Ausstehende vollständige Bearbeitungszeit von Operation O_{ipl} an Station M_k
$t_{ipl,k,\text{Rest}}$	Restliche Bearbeitungszeit von Operation O_{ipl} an Station M_k
$t_{ipl,\text{Rest}}$	Restliche Bearbeitungszeit der aktuellen Operation O_{ipl} von Auftrag J_i
$t_{i'p'l',\text{Rest}}$	Restliche Bearbeitungszeit der aktuellen Operation $O_{i'p'l'}$ auf dem Bearbeitungsplatz
$t_{i,\text{verbl}}$	Verbleibende Prozesszeit des aktuellen Prozesses von Auftrag J_i
t_{jetzt}	Aktueller Zeitpunkt
T_{max}	Maximale Zeit für Entscheidungsfindung
t_{opt}	Zeitlimit für die Optimierung
t_{PP}	Fließbandbedingte Transportzeit pro Pufferplatz
$t_{\text{proc,total},i}$	Gesamte Bearbeitungszeit eines Agenten i
t_s	Scheduling-Zeitpunkt
$t_{\text{stoer,ende}}$	Störungsende
$t_{\text{stoer,start}}$	Störungsstart
$t_{\text{sys},i}$	Aufenthaltszeit eines Agenten i in der Produktion
TTR	Time To Restoration
$t_{\text{TR},ipl}$	Transportzeit vor Operation O_{ipl}
$t_{\text{TR},max}$	Größtmögliche Transportzeit
$t_{\text{TR},x\to y}$	Transportzeit von x nach y
$t_{\text{TR},x\to y,\text{Rest}}$	Rest-Transportzeit von x nach y
$t_{u,\text{ende}}$	Frühestmögliche Endzeit des Knotens u
$t_{u,\text{start}}$	Frühestmögliche Startzeit des Knotens u
$t'_{u,\text{start}}$	Tatsächliche Startzeit des Knotens u
$t_{\text{warte},i}$	Wartezeit eines Agenten i im Ausgangspuffer
$t_{\text{WZ},ipl}$	Aufenthaltszeit im Eingangspuffer vor Operation O_{ipl}
t_{zyk}	Theoretische Zykluszeit der Produktion
U	Menge an Knoten u
u	Knoten aus der Menge U
UEZ_{ipl}	Übergangszeit eines Vorgangs O_{ipl}

$UEZ_{ipl,max}$	Größtmögliche Übergangszeit
$UEZ_{ipl,M_{k1}\rightarrow M_{k2},min}$	Minimal mögliche Übergangszeit zwischen den Stationen M_{k1} und M_{k2} für Operation O_{ipl}
$UEZ_{J,m}$	Mittlere Übergangszeit aller Aufträge in J
u_i	Nutzen des Agenten i
V	Menge Fahrerloser Transportfahrzeuge V_j
va_z	State-Information zu Actions A_z
v_i	Bewertung des Agenten i
V_j	Fahrzeug aus der Menge V
V_M	Stationsverfügbarkeit
$v_\pi(s)$	Value-Funktion (Wert) von State s für Policy π
vpt_x	State-Information zu Produkttyp PT_x auf Fahrzeug
vp_y	State-Information zu Ort des Fahrzeugs y
$\hat{v}(s_t)$	Wahrer Zielwert des Values für State s_t
v_V	Fahrzeuggeschwindigkeit
w	Wurzel des Wurzelbaums
W_i	Zulässiger Wertebereich eines Gens eines Individuums, welches einem Auftrag J_i zugeordnet ist
WT	Menge an Werkstückträgern WT_o
WT_o	Werkstückträger aus der Menge WT
X	Menge aller gültigen Allokationen
$\boldsymbol{x}(\boldsymbol{b})$	Vektor der Zuweisungen x_i der Gebote $\boldsymbol{b}$
x_i	Zuweisung des versteigerten Guts an Agent i

Literatur

[1] T. Bauernhansl, „Die Vierte Industrielle Revolution – Der Weg in ein wertschaffendes Produktionsparadigma,“ in *Industrie 4.0 in Produktion, Automatisierung und Logistik*, T. Bauernhansl, M. ten Hompel und B. Vogel-Heuser, Hrsg., Wiesbaden: Springer Fachmedien, 2014, S. 5–35.

[2] H. Kagermann, W.-D. Lukas und W. Wahlster, „Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution,“ *VDI nachrichten*, Jg. 13, Nr. 1, S. 2–3, 2011.

[3] D. Siepmann und N. Graef, „Industrie 4.0 - Grundlagen und Gesamtzusammenhang,“ in *Einführung und Umsetzung von Industrie 4.0*, A. Roth, Hrsg., Berlin, Heidelberg: Springer, 2016, S. 17–82.

[4] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld und M. Hoffmann, „Industrie 4.0,“ *Wirtschaftsinformatik*, Jg. 56, Nr. 4, S. 261–264, 2014.

[5] N. Boysen, M. Fliedner und A. Scholl, „Sequencing mixed-model assembly lines: Survey, classification and model critique,“ *European Journal of Operational Research*, Jg. 192, Nr. 2, S. 349–373, 2009.

[6] N. Boysen, „Produktionsplanung bei Variantenfließfertigung,“ *Operations Research Proceedings*, S. 11–15, 2006.

[7] N. Boysen, „Variantenfließfertigung,“ Dissertation, Universität Hamburg, 2005.

[8] J. Golz, „Materialbereitstellung bei Variantenfließlinien in der Automobilendmontage,“ Dissertation, Technische Universität Berlin, 2014.

[9] X. Zhu, S. J. Hu, Y. Koren und N. Huang, „A complexity model for sequence planning in mixed-model assembly lines,“ *Journal of Manufacturing Systems*, Jg. 31, Nr. 2, S. 121–130, 2012.

[10] Y. Koren, *The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2010.

[11] EU, „Regulation (EU) 2019/631 of the European Parliament and of the Council of 17 April 2019 setting CO2 emission performance standards for new passenger cars and for new light commercial vehicles, and repealing Regulations (EC) No 443/2009 and (EU) No 510/2011,“ *Official Journal of the European Union*, Jg. 111, S. 13–53, 2019.

[12] C. Thies, C. Hüls, K. Kieckhäfer, J. Wansart und T. S. Spengler, „Project portfolio planning under CO2 fleet emission restrictions in the automotive industry,“ *Journal of Industrial Ecology*, 2022.

[13] W. Kern, F. Rusitschka, W. Kopytynski, S. Keckl und T. Bauernhansl, „Alternatives to assembly line production in the automotive industry,“ in *Proceedings of the 23rd international conference on production research (IFPR)*, 2015.

[14] P. Foith-Förster, J.-H. Eising und T. Bauernhansl, „Effiziente Montagesysteme ohne Band und Takt,“ *wt Werkstattstechnik online*, Jg. 107, S. 169–175, 2017.

[15] S. J. Hu, J. Ko, L. Weyand, H. A. ElMaraghy, T. K. Lien, Y. Koren, H. Bley, G. Chryssolouris, N. Nasr und M. Shpitalni, „Assembly system design and operations for product variety,“ *CIRP Annals*, Jg. 60, Nr. 2, S. 715–733, 2011.

[16] P. Greschke, „Matrix-Produktion als Konzept einer taktunabhängigen Fließfertigung,“ Dissertation, Technische Universität Braunschweig, 2016.

[17] L. S. Bochmann, „Entwicklung und Bewertung eines flexiblen und dezentral gesteuerten Fertigungssystems für variantenreiche Produkte,“ Dissertation, ETH Zürich, 2018.

[18] W. Kern, „Modulare Produktion: Methodik zur Gestaltung eines modularen Montagesystems für die variantenreiche Serienmontage im Automobilbau,“ Dissertation, Universität Stuttgart, 2021.

[19] Verein Deutscher Ingenieure, *VDI 2510: Fahrerlose Transportsysteme (FTS)*, 2005.

[20] J. Milberg, G. Reinhart, D. Spath, M. Weck, G. Seliger und W. Eversheim et al., „Produktionssystemplanung,“ in *Produktion und Management 3*, W. Eversheim und G. Schuh, Hrsg., Berlin, Heidelberg: Springer, 1999, S. 118–244.

[21] B. Waschneck, T. Altenmüller, T. Bauernhansl und A. Kyek, „Production scheduling in complex job shops from an industrie 4.0 perspective: A review and challenges in the semiconductor industry,“ *1st International Workshop on Science, Application and Methods in Industry 4.0, SAMI 2016. Proceedings*, Jg. 2016,

[22] J. Browne, D. Dubois, K. Rathmill, S. P. Sethi und K. E. Stecke et al., „Classification of flexible manufacturing systems,“ *The FMS magazine*, Jg. 2, Nr. 2, S. 114–117, 1984.

[23] H. A. ElMaraghy, „Flexible and reconfigurable manufacturing systems paradigms,“ *International Journal of Flexible Manufacturing Systems*, Jg. 17, Nr. 4, S. 261–276, 2005.

[24] A. M. Newman und C. Arai Yano, „Centralized and decentralized train scheduling for intermodal operations,“ *IIE Transactions*, Jg. 32, Nr. 8, S. 743–754, 2000.

[25] M. Mitici, J. Goseling, M. de Graaf und R. J. Boucherie, „Decentralized vs. centralized scheduling in wireless sensor networks for data fusion," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, S. 5070–5074.

[26] A. Mehrsai, G. Figueira, N. Santos, P. Amorim und B. Almada-Lobo, „Decentralized Vs. Centralized Sequencing in a Complex Job-Shop Scheduling," in *Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing*, Ser. IFIP Advances in Information and Communication Technology, H. Lödding, R. Riedel, K.-D. Thoben, G. von Cieminski und D. Kiritsis, Hrsg., Bd. 513, Cham: Springer International Publishing, 2017, S. 467–474.

[27] R. F. Atallah, C. M. Assi, W. Fawaz, M. H. K. Tushar und M. J. Khabbaz, „Optimal Supercharge Scheduling of Electric Vehicles: Centralized Versus Decentralized Methods," *IEEE Transactions on Vehicular Technology*, Jg. 67, Nr. 9, S. 7896–7909, 2018.

[28] Audi AG, *Neue Ära: Audi Hungaria startet Serienproduktion von Elektromotoren. Pressemitteilung*, 24.07.2018. Adresse: `https://www.audi-mediacenter.com/de/pressemitteilungen/neue-aera-audi-hungaria-startet-serienproduktion-von-elektromotoren-10495` (besucht am 01. 05. 2022).

[29] S. Mayer und C. Endisch, „Adaptive Production Control in a Modular Assembly System Based on Partial Look-ahead Scheduling," *IEEE International Conference on Mechatronics 2019*, S. 293–300, 2019.

[30] S. Mayer, J. Schmidt und C. Endisch, „Steuereinheit sowie Verfahren zum Steuern von Transporten einer Vielzahl an Werkstücken in einem modularen Montagesystem mittels eines genetischen Algorithmus; Montagesystem," DPMA-Aktenzeichen 102020122194.0, (Beim Patentamt am 25.08.2020 eingereicht).

[31] S. Mayer, J. Schmidt und C. Endisch, „Verfahren sowie Steuereinheit zum automatisierten Einlasten jeweiliger Fertigungsprozesse von Werkstücken unterschiedlicher Typen in einem gewünschten Verhältnis," DPMA-Aktenzeichen 102020124684.6, (Beim Patentamt am 22.09.2020 eingereicht).

[32] S. Mayer, N. Hohme, D. Gankin und C. Endisch, „Adaptive Production Control in a Modular Assembly System – Towards an Agent-based Approach," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, IEEE, 2019, S. 45–52.

[33] S. Mayer, D. Gankin, C. Arnet und C. Endisch, „Adaptive Production Control with Negotiating Agents in Modular Assembly Systems," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE, 2019, S. 120–127.

[34] S. Mayer, T. Classen und C. Endisch, „Modular production control using deep reinforcement learning: proximal policy optimization,“ *Journal of Intelligent Manufacturing*, 2021.

[35] S. Mayer, T. Classen, J. Schmidt und C. Endisch, „Verfahren und Steuervorrichtung zur Erzeugung wenigstens eines Fahrauftrags für wenigstens ein fahrerloses Transportfahrzeug, FTF, eines modularen Montagesystems,“ DPMA-Aktenzeichen 102020127407.6, (Beim Patentamt am 19.10.2020 eingereicht).

[36] D. Gankin, S. Mayer, J. Zinn, B. Vogel-Heuser und C. Endisch, „Modular Production Control with Multi-Agent Deep Q-Learning,“ in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2021.

[37] S. Mayer, C. Arnet, D. Gankin und C. Endisch, „Standardized Framework for Evaluating Centralized and Decentralized Control Systems in Modular Assembly Systems,“ in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE, 2019, S. 113–119.

[38] H.-P. Wiendahl, *Betriebsorganisation für Ingenieure*, 1., überarbeitete Auflage. München: Hanser, 2014.

[39] H.-P. Wiendahl und H.-H. Wiendahl, *Betriebsorganisation für Ingenieure*, 9., vollständig überarbeitete Auflage. München: Hanser, 2019.

[40] W. Eversheim, *Organisation in der Produktionstechnik: Band 1: Grundlagen*, Dritte, neu bearbeitete und erweiterte Auflage, Ser. Studium und Praxis. Berlin, Heidelberg: Imprint und Springer, 1996.

[41] DIN Deutsches Institut für Normung, *DIN 8580: Fertigungsverfahren*, 2003.

[42] H.-P. Wiendahl, J. Reichardt und P. Nyhuis, *Handbuch Fabrikplanung: Konzept, Gestaltung und Umsetzung wandlungsfähiger Produktionsstätten*, 2., überarb. und erw. Aufl. München: Hanser, 2014.

[43] T. Bauernhansl, M. Fechter und T. Dietz, *Entwicklung, Aufbau und Demonstration einer wandlungsfähigen (Fahrzeug-) Forschungsproduktion*, 1st ed. 2020, Ser. ARENA2036. Berlin, Heidelberg: Springer und Springer Vieweg, 2020.

[44] T. Bauernhansl, „Automotive industry without conveyor belt and cycle – research campus ARENA2036,“ in *15. Internationales Stuttgarter Symposium*, Ser. Proceedings, M. Bargende, H.-C. Reuss und J. Wiedemann, Hrsg., Wiesbaden: Springer Vieweg, 2015, S. 347–356.

[45] D. Steegmüller und M. Zürn, „Wandlungsfähige Produktionssysteme für den Automobilbau der Zukunft,“ in *Industrie 4.0 in Produktion, Automatisierung und Logistik*, T. Bauernhansl, M. ten Hompel und B. Vogel-Heuser, Hrsg., Wiesbaden: Springer Fachmedien, 2014, S. 103–119.

[46] P. Foith-Förster und T. Bauernhansl, „Changeable and reconfigurable assembly systems: A structure planning approach in automotive manufacturing,“ in *15. Internationales Stuttgarter Symposium*, Ser. Proceedings, M. Bargende, H.-C. Reuss und J. Wiedemann, Hrsg., Wiesbaden: Springer Vieweg, 2015, S. 375–394.

[47] M. Hofmann, *Montage- und Logistik-FTF für die Automobilproduktion ohne Band und Takt*, 2016.

[48] J. Popp, *Neuartige Logistikkonzepte für eine flexible Automobilproduktion ohne Band*, 1. Auflage, Ser. Berichte aus dem Institut für Fördertechnik und Logistik. Berlin: epubli, 2019.

[49] M. ten Hompel, „Zellulare Fördertechnik,“ *Logistics Journal nicht-referierte Veröffentlichungen*, 2006.

[50] T. Kirks, J. Stenzel, A. Kamagaew und M. ten Hompel, „Zellulare Transportfahrzeuge für flexible und wandelbare Intralogistiksysteme,“ *Logistics Journal Proceedings*, 2012.

[51] L. Bochmann, L. Gehrke, N. Gehrke, C. Mertens und O. Seiss, „Innovative Konzepte einer sich selbstorganisierenden Fahrzeugmontage am Beispiel des Forschungsprojekts SMART FACE,“ in *Einführung und Umsetzung von Industrie 4.0*, A. Roth, Hrsg., 2016, S. 173–191.

[52] A. Böckenkamp, C. Mertens, C. Prasse, J. Stenzel und F. Weichert, „A Versatile and Scalable Production Planning and Control System for Small Batch Series,“ in *Industrial internet of things*, Ser. Springer series in wireless technology, S. Jeschke, C. Brecher und H. Song, Hrsg., 2017, S. 541–559.

[53] D. Kupzik, S. Haas, D. Stemler, J. Pfeifle und J. Fleischer, „Development of a manufacturing cell for fixtureless joining in car body assembly,“ in *Tagungsband des 3. Kongresses Montage Handhabung Industrieroboter*, T. Schüppstuhl, K. Tracht und J. Franke, Hrsg., Berlin, Heidelberg: Springer, 2018, S. 23–30.

[54] F. Echsler Minguillon und G. Lanza, „Maschinelles Lernen in der PPS,“ *wt Werkstattstechnik online*, Jg. 107, Nr. 9, S. 630–634, 2017.

[55] F. Echsler Minguillon und G. Lanza, „Coupling of centralized and decentralized scheduling for robust production in agile production systems,“ *Procedia CIRP*, Jg. 79, S. 385–390, 2019.

[56] R. Schmitt, A. Göppert, G. Hüttemann, P. Lettmann, K. Rook-Weiler, D. Schönstein, A. Schreiber, E. Serbest, M. Steffens und A. Tomys-Brummerloh, „Frei verkettete wandlungsfähige Montage,“ in *Internet of production für agile Unternehmen*, C. Brecher, F. Klocke und R. Schmitt, Hrsg., 2017, S. 339–368.

[57] A. Göppert, G. Hüttemann, S. Jung, D. Grunert und R. Schmitt, „Frei verkettete Montagesysteme,“ *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, Jg. 113, Nr. 3, S. 151–155, 2018.

[58] G. Hüttemann, A. F. Buckhorst und R. H. Schmitt, „Modelling and Assessing Lineless Mobile Assembly Systems,“ *Procedia CIRP*, Jg. 81, S. 724–729, 2019.

[59] D. Grunert, S. Jung, T. Leipold, A. Göppert, G. Hüttemann und R. Schmitt, „Service-oriented Information Model for the Model-Driven Control of Dynamically Interconnected Assembly Systems,“ in *Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter*, T. Schüppstuhl, K. Tracht und J. Roßmann, Hrsg., 2019, S. 23–33.

[60] A. Göppert, J. Rachner und R. H. Schmitt, „Automated scenario analysis of reinforcement learning controlled line-less assembly systems,“ *Procedia CIRP*, Jg. 93, S. 1091–1096, 2020.

[61] F. Hees, S. Müller-Abdelrazeq, M. Voss, R. Schmitt, G. Hüttemann und K. Rook-Weiler et al., Hrsg., *Projektatlas Kompetenz Montage: Kollaborativ und wandlungsfähig.* Aachen: RWTH Aachen University, Lehrstuhl für Informationsmanagement im Maschinenbau, 2019.

[62] P. Lettmann, G. Hüttemann und R. Schmitt, „Produktrouten in frei verketteten Montagesystemen,“ *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, Jg. 114, Nr. 9, S. 517–520, 2019.

[63] W. Kern, F. Rusitschka und T. Bauernhansl, „Planning of Workstations in a Modular Automotive Assembly System,“ *Procedia CIRP*, Jg. 57, S. 327–332, 2016.

[64] W. Kern, H. Lämmermann und T. Bauernhansl, „An Integrated Logistics Concept for a Modular Assembly System,“ *Procedia Manufacturing*, Jg. 11, S. 957–964, 2017.

[65] H. Borkeloh, „Montage ohne Band und Takt: KUKA Matrixproduktion,“ in *1. Stuttgarter Tagung zur Zukunft der Automobilproduktion*, M. Fechter, D. Korte und J. Henke, Hrsg., 2019.

[66] W. Hackenberg und T. Gagern, „Intralogistik - Ökosystem für die Fabrik der Zukunft,“ in *11. Sick Automobiltage*, 2019.

[67] S. T. Newman und A. Nassehi, „Machine tool capability profile for intelligent process planning,“ *CIRP Annals*, Jg. 58, Nr. 1, S. 421–424, 2009.

[68] K. Dorofeev und A. Zoitl, „Skill-based Engineering Approach using OPC UA Programs,“ in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, IEEE, 2018, S. 1098–1103.

[69] A. Perzylo, J. Grothoff, L. Lucio, M. Weser, S. Malakuti, P. Venet, V. Aravantinos und T. Deppe, „Capability-based semantic interoperability of manufacturing resources: A BaSys 4.0 perspective," *IFAC-PapersOnLine*, Jg. 52, Nr. 13, S. 1590–1596, 2019.

[70] K. Watermeyer, *Ablaufplanung mit alternativen Prozessplänen*, Ser. BestMasters. Wiesbaden: Springer Gabler, 2016.

[71] Y. K. Kim, K. Park und J. Ko, „A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Computers and Operations Research*, Jg. 30, Nr. 8, S. 1151–1171, 2003.

[72] P. Nyhuis und H.-P. Wiendahl, *Logistische Kennlinien*. Berlin, Heidelberg: Springer, 2012.

[73] E. Gutenberg, *Grundlagen der Betriebswirtschaftslehre*. Berlin, Heidelberg: Springer, 1951.

[74] H. Lödding, *Verfahren der Fertigungssteuerung: Grundlagen, Beschreibung, Konfiguration*, 3. Auflage, Ser. VDI-Buch. Berlin, Heidelberg: Springer Vieweg, 2016.

[75] D. Ouelhadj und S. Petrovic, „A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, Jg. 12, Nr. 4, S. 417–431, 2009.

[76] A. Birolini, *Zuverlässigkeit von Geräten und Systemen*. Berlin, Heidelberg: Springer, 1997.

[77] J. R. Müller, *Die Formalisierte Terminologie der Verlässlichkeit Technischer Systeme*. Berlin, Heidelberg: Springer, 2015.

[78] J. D. C. Little, „A Proof for the Queuing Formula," *Operations Research*, Jg. 9, Nr. 3, S. 383–387, 1961.

[79] M. Schotten, „Grundlagen der Produktionsplanung und -steuerung," in *Produktionsplanung und -steuerung*, Ser. VDI-Buch, H. Luczak und W. Eversheim, Hrsg., Berlin, Heidelberg: Springer, 1999.

[80] R. Hackstein, *Produktionsplanung und -steuerung (PPS): Ein Handbuch für die Betriebspraxis*, 2., überarb. Aufl. Düsseldorf: VDI-Verl., 1989.

[81] A.-W. Scheer, *Wirtschaftsinformatik Studienausgabe: Referenzmodelle für industrielle Geschäftsprozesse*. Berlin, Heidelberg: Imprint und Springer, 1995.

[82] G. Schuh und V. Stich, *Produktionsplanung und -steuerung*, 4., überarbeitete Auflage. Berlin: Springer, 2012.

[83] D. Weidner, „Engpaßorientierte Fertigungssteuerung: Eine Untersuchung über die in optimized production technology implementierten Konzepte der Produktionsplanung und -steuerung," Dissertation, Universität Kiel, 1992.

[84] W. Bechte, „Steuerung der Durchlaufzeit durch belastungsorientierte Auftragsfreigabe bei Werkstattfertigung,“ Dissertation, Universität Hannover, 1984.

[85] H.-P. Wiendahl, *Fertigungsregelung: Logistische Beherrschung von Fertigungsabläufen auf Basis des Trichtermodells*. München: Hanser, 1997.

[86] M. L. Spearman, D. L. Woodruff und W. J. Hopp, „CONWIP: a pull alternative to kanban,“ *International Journal of Production Research*, Jg. 28, Nr. 5, S. 879–894, 1990.

[87] A. D. Baker, „A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: Dispatching, scheduling, and pull,“ *Journal of Manufacturing Systems*, Jg. 17, Nr. 4, S. 297–320, 1998.

[88] P. Leitão, „Agent-based distributed manufacturing control: A state-of-the-art survey,“ *Engineering Applications of Artificial Intelligence*, Jg. 22, Nr. 7, S. 979–991, 2009.

[89] M. ten Hompel und V. Heidenblut, *Taschenlexikon Logistik: Abkürzungen, Definitionen und Erläuterungen der wichtigsten Begriffe aus Materialfluss und Logistik*, Ser. VDI-Buch. Berlin, Heidelberg: Springer, 2011.

[90] M. Pinedo, *Scheduling: Theory, algorithms, and systems*, Fifth Edition. Cham u. a.: Springer, 2016.

[91] C. Özgüven, L. Özbakır und Y. Yavuz, „Mathematical models for job-shop scheduling problems with routing and process plan flexibility,“ *Applied Mathematical Modelling*, Jg. 34, Nr. 6, S. 1539–1548, 2010.

[92] Y. Crama, „Combinatorial optimization models for production scheduling in automated manufacturing systems,“ *European Journal of Operational Research*, Jg. 99, Nr. 1, S. 136–153, 1997.

[93] K. Mehlhorn und P. Sanders, *Algorithms and data structures: The basic toolbox*. Berlin: Springer, 2008.

[94] M. R. Garey, D. S. Johnson und R. Sethi, „The Complexity of Flowshop and Jobshop Scheduling,“ *Mathematics of Operations Research*, Jg. 1, Nr. 2, S. 117–129, 1976.

[95] J. D. Ullman, „NP-complete scheduling problems,“ *Journal of Computer and System Sciences*, Jg. 10, Nr. 3, S. 384–393, 1975.

[96] R. L. Graham, E. L. Lawler, J. K. Lenstra und A. Kan, „Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey,“ in *Annals of Discrete Mathematics*, Bd. 5, Elsevier, 1979, S. 287–326.

[97] B. L. MacCarthy und J. Liu, „Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling,“ *International Journal of Production Research*, Jg. 31, Nr. 1, S. 59–79, 1993.

[98] P. Cowling und M. Johansson, „Using real time information for effective dynamic scheduling,“ *European Journal of Operational Research*, Jg. 139, Nr. 2, S. 230–244, 2002.

[99] V. Suresh und D. Chaudhuri, „Dynamic scheduling—a survey of research,“ *International Journal of Production Economics*, Jg. 32, Nr. 1, S. 53–63, 1993.

[100] J. Mohan, K. Lanka und A. N. Rao, „A Review of Dynamic Job Shop Scheduling Techniques,“ *Procedia Manufacturing*, Jg. 30, S. 34–39, 2019.

[101] D. A. Rossit, F. Tohmé und M. Frutos, „Production planning and scheduling in Cyber-Physical Production Systems: a review,“ *International Journal of Computer Integrated Manufacturing*, Jg. 32, Nr. 4-5, S. 385–395, 2019.

[102] M. Mousavi, H. J. Yap, S. N. Musa, F. Tahriri und S. Z. Md Dawal, „Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization,“ *PloS one*, Jg. 12, Nr. 3, e0169817, 2017.

[103] H. Hwang und S. H. Kim, „Development of dispatching rules for automated guided vehicle systems,“ *Journal of Manufacturing Systems*, Jg. 17, Nr. 2, S. 137–143, 1998.

[104] Y. Zheng, Y. Xiao und Y. Seo, „A tabu search algorithm for simultaneous machine/AGV scheduling problem,“ *International Journal of Production Research*, Jg. 52, Nr. 19, S. 5748–5763, 2014.

[105] U. A. Umar, M. K. A. Ariffin, N. Ismail und S. H. Tang, „Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (AGV) in flexible manufacturing systems (FMS) environment,“ *The International Journal of Advanced Manufacturing Technology*, Jg. 81, Nr. 9-12, S. 2123–2141, 2015.

[106] T. Miyamoto und K. Inoue, „Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems,“ *Computers and Industrial Engineering*, Jg. 91, S. 1–9, 2016.

[107] N. Smolic-Rocak, S. Bogdan, Z. Kovacic und T. Petrovic, „Time Windows Based Dynamic Routing in Multi-AGV Systems,“ *IEEE Transactions on Automation Science and Engineering*, Jg. 7, Nr. 1, S. 151–155, 2010.

[108] L. Qiu, W.-J. Hsu, S.-Y. Huang und H. Wang, „Scheduling and routing algorithms for AGVs: A survey,“ *International Journal of Production Research*, Jg. 40, Nr. 3, S. 745–760, 2002.

[109] H. E. Nouri, O. B. Driss und K. Ghédira, „A Classification Schema for the Job Shop Scheduling Problem with Transportation Resources: State-of-the-Art Review,“ in *Silhavy, Senkerik et al. (Hg.) 2016 – Artificial Intelligence Perspectives in Intelligent Systems*, Bd. 464, 2016, S. 1–11.

[110] K.-P. Kistner und M. Steven, *Produktionsplanung*. Heidelberg: Physica-Verlag HD, 2001.

[111] I. Kacem, S. Hammadi und P. Borne, „Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, Jg. 32, Nr. 1, S. 1–13, 2002.

[112] A. Bożek und M. Wysocki, „Off-Line and Dynamic Production Scheduling – A Comparative Case Study," *Management and Production Engineering Review*, Jg. 7, Nr. 1, S. 21–32, 2016.

[113] B. Vogel-Heuser, P. Göhner und A. Lüder, „Agent-based Control of Production System—and its Architectural Challenges," in *Industrial agents*, P. Leitao und S. Karnouskos, Hrsg., Amsterdam, Netherlands: Elsevier, 2015, S. 153–170.

[114] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord und J. Stafford, *Documenting Software Architectures: Views and Beyond, Second Edition*, 2nd edition. Addison-Wesley Professional, 2011.

[115] D. M. Dilts, N. P. Boyd und H. H. Whorms, „The evolution of control architectures for automated manufacturing systems," *Journal of Manufacturing Systems*, Jg. 10, Nr. 1, S. 79–93, 1991.

[116] R. Unland, „Industrial Agents," in *Industrial agents*, P. Leitao und S. Karnouskos, Hrsg., Amsterdam, Netherlands: Elsevier, 2015, S. 23–44.

[117] I. A. Chaudhry und A. A. Khan, „A research survey: review of flexible job shop scheduling techniques," *International Transactions in Operational Research*, Jg. 23, Nr. 3, S. 551–591, 2016.

[118] M. Godinho Filho, C. F. Barco und R. F. Tavares Neto, „Using Genetic Algorithms to solve scheduling problems on flexible manufacturing systems (FMS): a literature survey, classification and analysis," *Flexible Services and Manufacturing Journal*, Jg. 26, Nr. 3, S. 408–431, 2014.

[119] I. Gerdes, F. Klawonn und R. Kruse, *Evolutionäre Algorithmen: Genetische Algorithmen – Strategien und Optimierungsverfahren – Beispielanwendungen*, Ser. Computational Intelligence. Wiesbaden: Vieweg+Teubner Verlag, 2004.

[120] V. Mařík und J. Lažanský, „Industrial applications of agent technologies," *Control Engineering Practice*, Jg. 15, Nr. 11, S. 1364–1380, 2007.

[121] T. Borangiu, „A Service-Orientated Arhitecture for Holonic Manufacturing Control," in *Towards Intelligent Engineering and Information Technology*, Ser. Studies in Computational Intelligence, J. Kacprzyk, I. J. Rudas und J. Fodor, Hrsg., Bd. 243, Berlin, Heidelberg: Springer, 2009, S. 489–503.

[122] D. Trentesaux, „Distributed control of production systems,“ *Engineering Applications of Artificial Intelligence*, Jg. 22, Nr. 7, S. 971–978, 2009.

[123] Y. Sallez, T. Berger, S. Raileanu, S. Chaabane und D. Trentesaux, „Semi-heterarchical control of FMS: From theory to application,“ *Engineering Applications of Artificial Intelligence*, Jg. 23, Nr. 8, S. 1314–1326, 2010.

[124] W. Shen, Q. Hao, H. J. Yoon und D. H. Norrie, „Applications of agent-based systems in intelligent manufacturing: An updated review,“ *Advanced Engineering Informatics*, Jg. 20, Nr. 4, S. 415–431, 2006.

[125] L. Monostori, J. Váncza und S. Kumara, „Agent-Based Systems for Manufacturing,“ *CIRP Annals*, Jg. 55, Nr. 2, S. 697–720, 2006.

[126] M. Metzger und G. Polakow, „A Survey on Applications of Agent Technology in Industrial Process Control,“ *IEEE Transactions on Industrial Informatics*, Jg. 7, Nr. 4, S. 570–581, 2011.

[127] S. Wang, J. Wan, D. Zhang, Di Li und C. Zhang, „Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination,“ *Computer Networks*, Jg. 101, S. 158–168, 2016.

[128] P. Leitão, N. Rodrigues, J. Barbosa, C. Turrin und A. Pagani, „Intelligent products: The grace experience,“ *Control Engineering Practice*, Jg. 42, S. 95–105, 2015.

[129] L. D. Xu, E. L. Xu und L. Li, „Industry 4.0: state of the art and future trends,“ *International Journal of Production Research*, Jg. 56, Nr. 8, S. 2941–2962, 2018.

[130] F. Wortmann und K. Flüchter, „Internet of Things,“ *Business & Information Systems Engineering*, Jg. 57, Nr. 3, S. 221–224, 2015.

[131] B. Hayes, „Cloud computing,“ *Communications of the ACM*, Jg. 51, Nr. 7, S. 9–11, 2008.

[132] L. Monostori, „Cyber-physical Production Systems: Roots, Expectations and R&D Challenges,“ *Procedia CIRP*, Jg. 17, S. 9–13, 2014.

[133] Acatech, *Cyber-Physical Systems: Innovationsmotor für Mobilität, Gesundheit, Energie und Produktion*. Berlin, Heidelberg: Springer, 2011.

[134] CPS-VO, *Strategic R&D Opportunities for 21st Century Cyber-Physical Systems*, Cyber-Physical Systems Virtual Organization, Hrsg., 2013.

[135] K.-J. Park, R. Zheng und X. Liu, „Cyber-physical systems: Milestones and research challenges,“ *Computer Communications*, Jg. 36, Nr. 1, S. 1–7, 2012.

[136] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn und K. Ueda, „Cyber-physical systems in manufacturing,“ *CIRP Annals*, Jg. 65, Nr. 2, S. 621–641, 2016.

[137] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE, *Integration von Unternehmensführungs- und Leitsystemen*, 2014.

[138] B. Vogel-Heuser, G. Kegel, K. Bender und K. Wucherer, „Global Information Architecture for Industrial Automation," *atp magazin*, Jg. 51, Nr. 01-02, S. 108–115, 2013.

[139] T. Heimbold, *Einführung in die Automatisierungstechnik: Automatisierungssysteme, Komponenten, Projektierung und Planung*. München: Carl Hanser Verlag GmbH & Co. KG, 2014.

[140] D. A. Rossit, F. Tohmé und M. Frutos, „Industry 4.0: Smart Scheduling," *International Journal of Production Research*, Jg. 57, Nr. 12, S. 3802–3813, 2019.

[141] D. Rossit und F. Tohmé, „Scheduling research contributions to Smart manufacturing," *Manufacturing Letters*, Jg. 15, S. 111–114, 2018.

[142] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee und B. Yin, „Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges," *IEEE Access*, Jg. 6, S. 6505–6519, 2017.

[143] Y. Lu, „Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, Jg. 6, S. 1–10, 2017.

[144] R. Y. Zhong, X. Xu, E. Klotz und S. T. Newman, „Intelligent Manufacturing in the Context of Industry 4.0: A Review," *Engineering*, Jg. 3, Nr. 5, S. 616–630, 2017.

[145] E. Oztemel und S. Gursev, „Literature review of Industry 4.0 and related technologies," *Journal of Intelligent Manufacturing*, Jg. 31, Nr. 1, S. 127–182, 2020.

[146] B. Vogel-Heuser, J. Lee und P. Leitão, „Agents enabling cyber-physical production systems," *at - Automatisierungstechnik*, Jg. 63, Nr. 10, 2015.

[147] M. J. Wooldridge, *An introduction to multiagent systems*, 2. ed. Chichester: Wiley, 2009.

[148] M. J. Wooldridge und N. R. Jennings, „Intelligent agents: Theory and practice," *The knowledge engineering review*, Jg. 10, Nr. 2, S. 115–152, 1995.

[149] A. Dorri, S. S. Kanhere und R. Jurdak, „Multi-Agent Systems: A Survey," *IEEE Access*, Jg. 6, S. 28 573–28 593, 2018.

[150] R. Unland, „Software agent systems," in *Industrial Agents*, Elsevier, 2015, S. 3–22.

[151] M. Wooldridge, „Agent-based software engineering," *IEE Proceedings - Software Engineering*, Jg. 144, Nr. 1, S. 26, 1997.

[152] V. Mařík, M. Fletcher und M. Pěchouček, „Holons & Agents: Recent Developments and Mutual Impacts,“ in *Multi-Agent Systems and Applications II*, Ser. Lecture Notes in Computer Science, G. Goos, J. Hartmanis, J. van Leeuwen, V. Mařík, O. Štěpánková, H. Krautwurmová und M. Luck, Hrsg., Bd. 2322, Berlin, Heidelberg: Springer, 2002, S. 233–267.

[153] S. Poslad, „Specifying protocols for multi-agent systems interaction,“ *ACM Transactions on Autonomous and Adaptive Systems*, Jg. 2, Nr. 4, 15–es, 2007.

[154] A. Saad, G. Biswas, K. Kawamura und E. M. Johnson, „Effectiveness of dynamic rescheduling in agent-based flexible manufacturing systems,“ in *Architectures, Networks, and Intelligent Systems for Manufacturing Integration*, B. Gopalakrishnan, S. Murugesan, O. Struger und G. Zeichen, Hrsg., Ser. SPIE Proceedings, SPIE, 1997, S. 88–99.

[155] W. Shen und D. H. Norrie, „An agent-based approach for dynamic manufacturing scheduling,“ in *Proceedings of Workshop on Agent-Based Manufacturing*, Citeseer, 1998, S. 117–128.

[156] D. Ouelhadj, P. Cowling und S. Petrovic, „Contract net protocol for cooperative optimisation and dynamic scheduling of steel production,“ in *Intelligent systems design and applications*, Springer, 2003, S. 469–478.

[157] P. Leitão und F. Restivo, „A holonic approach to dynamic manufacturing scheduling,“ *Robotics and Computer-Integrated Manufacturing*, Jg. 24, Nr. 5, S. 625–634, 2008.

[158] K. Wang und S. Choi, „A holonic approach to flexible flow shop scheduling under stochastic processing times,“ *Computers and operations research*, Jg. 43, S. 157–168, 2014.

[159] Y. Liu, L. Wang, Y. Wang, X. V. Wang und L. Zhang, „Multi-agent-based scheduling in cloud manufacturing with dynamic task arrivals,“ *Procedia CIRP*, Jg. 72, S. 953–960, 2018.

[160] H. V. D. Parunak, „Manufacturing experience with the contract net,“ *Distributed artificial intelligence*, Jg. 1, S. 285–310, 1987.

[161] Smith, „The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver,“ *IEEE Transactions on Computers*, Jg. C-29, Nr. 12, S. 1104–1113, 1980.

[162] R. Davis und R. G. Smith, „Negotiation as a metaphor for distributed problem solving,“ *Artificial intelligence*, Jg. 20, Nr. 1, S. 63–109, 1983.

[163] A. Koestler, *The ghost in the machine.* Macmillan, 1968.

[164] H. van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts und P. Peeters, „Reference architecture for holonic manufacturing systems: PROSA," *Computers in Industry*, Jg. 37, Nr. 3, S. 255–274, 1998.

[165] P. Leitão und F. Restivo, „ADACOR: A holonic architecture for agile and adaptive manufacturing control," *Computers in Industry*, Jg. 57, Nr. 2, S. 121–130, 2006.

[166] L. Mönch, M. Stehli und J. Zimmermann, „FABMAS: An Agent-Based System for Production Control of Semiconductor Manufacturing Processes," in *Holonic and Multi-Agent Systems for Manufacturing*, Ser. Lecture Notes in Computer Science, G. Goos, J. Hartmanis, J. van Leeuwen, V. Mařík, D. McFarlane und P. Valckenaers, Hrsg., Bd. 2744, Berlin, Heidelberg: Springer, 2003, S. 258–267.

[167] O. Cardin, D. Trentesaux, A. Thomas, P. Castagna, T. Berger und H. Bril El-Haouzi, „Coupling predictive scheduling and reactive control in manufacturing hybrid control architectures: state of the art and future challenges," *Journal of Intelligent Manufacturing*, Jg. 28, Nr. 7, S. 1503–1517, 2017.

[168] W. Derigent, O. Cardin und D. Trentesaux, „Industry 4.0: contributions of holonic manufacturing control architectures and future challenges," *Journal of Intelligent Manufacturing*, 2020.

[169] J. Barbosa, P. Leitão, E. Adam und D. Trentesaux, „Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution," *Computers in Industry*, Jg. 66, S. 99–111, 2015.

[170] T. Borangiu, S. Răileanu, T. Berger und D. Trentesaux, „Switching mode control strategy in manufacturing execution systems," *International Journal of Production Research*, Jg. 53, Nr. 7, S. 1950–1963, 2015.

[171] Y. Shoham und K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge: Cambridge University Press, 2009.

[172] L. Buşoniu, R. Babuška und B. de Schutter, „Multi-agent Reinforcement Learning: An Overview," in *Innovations in Multi-Agent Systems and Applications - 1*, Ser. Studies in Computational Intelligence, J. Kacprzyk, D. Srinivasan und L. C. Jain, Hrsg., Bd. 310, Berlin, Heidelberg: Springer, 2010, S. 183–221.

[173] I. Goodfellow, Y. Bengio und A. Courville, *Deep Learning*. MIT Press, 2016.

[174] T. Gabel, „Multi-agent reinforcement learning approaches for distributed job-shop scheduling problems," Dissertation, Universität Osnabrück, 2009.

[175] N. Stricker, A. Kuhnle, R. Sturm und S. Friess, „Reinforcement learning for adaptive order dispatching in the semiconductor industry," *CIRP Annals*, Jg. 67, Nr. 1, S. 511–514, 2018.

[176] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra und M. Riedmiller, „Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, 2013.

[177] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel und D. Hassabis, „Mastering the game of Go with deep neural networks and tree search," *Nature*, Jg. 529, Nr. 7587, S. 484–489, 2016.

[178] R. S. Sutton und A. Barto, *Reinforcement learning: An introduction*, Second edition, Ser. Adaptive computation and machine learning. Cambridge, MA und London: The MIT Press, 2018.

[179] M. Morales, *Grokking Deep Reinforcement Learning*. O'Reilly Media, 2020.

[180] S. Vaidya, P. Ambad und S. Bhosle, „Industry 4.0 – A Glimpse," *Procedia Manufacturing*, Jg. 20, S. 233–238, 2018.

[181] M. Schluse, M. Priggemeyer, L. Atorf und J. Rossmann, „Experimentable Digital Twins—Streamlining Simulation-Based Systems Engineering for Industry 4.0," *IEEE Transactions on Industrial Informatics*, Jg. 14, Nr. 4, S. 1722–1731, 2018.

[182] A. G. Frank, L. S. Dalenogare und N. F. Ayala, „Industry 4.0 technologies: Implementation patterns in manufacturing companies," *International Journal of Production Economics*, Jg. 210, S. 15–26, 2019.

[183] G. Reinhart und G. Wünsch, „Economic application of virtual commissioning to mechatronic production systems," *Production Engineering*, Jg. 1, Nr. 4, S. 371–379, 2007.

[184] Verein Deutscher Ingenieure, *VDI/VDE 3693: Virtuelle Inbetriebnahme*, 2016.

[185] S. T. Mortensen und O. Madsen, „A Virtual Commissioning Learning Platform," *Procedia Manufacturing*, Jg. 23, S. 93–98, 2018.

[186] S. Seidel, U. Donath und J. Haufe, „Towards an integrated simulation and virtual commissioning environment for controls of material handling systems," in *Proceedings of the 2012 Winter Simulation Conference*, 2012, S. 1–12.

[187] M. Schleipen, O. Sauer, N. Friess, L. Braun und K. Shakerian, „Production Monitoring and Control Systems within the Digital Factory," in *Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology*, Ser. Advances in Intelligent and Soft Computing, J. Kacprzyk, G. Q. Huang, K. L. Mak und P. G. Maropoulos, Hrsg., Bd. 66, Berlin, Heidelberg: Springer, 2010, S. 711–724.

[188] M. F. Zäh, U. Munzert und G. Wünsch, „Simulation von Montageanlagen in der Automobilindustrie,“ *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, Jg. 99, Nr. 1-2, S. 18–21, 2004.

[189] L. Mönch, O. Rose und R. Sturm, „A Simulation Framework for the Performance Assessment of Shop-Floor Control Systems,“ *SIMULATION*, Jg. 79, Nr. 3, S. 163–170, 2003.

[190] A. Negahban und J. S. Smith, „Simulation for manufacturing system design and operation: Literature review and analysis,“ *Journal of Manufacturing Systems*, Jg. 33, Nr. 2, S. 241–261, 2014.

[191] J. W. Fowler und O. Rose, „Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems,“ *SIMULATION*, Jg. 80, Nr. 9, S. 469–476, 2004.

[192] S. Schreiber, „Entwicklung einer Vergleichs- und Bewertungsmöglichkeit von dezentralen Steuerungsarchitekturen für Produktionssysteme,“ Dissertation, Universität der Bundeswehr Hamburg, 2013.

[193] J. Sundermeier, F. Gehlhoff und A. Fay, „Development of a Simulation Model to Analyze the Performance of Decentral Rescheduling Algorithms in Production Systems,“ *SNE Simulation Notes Europe*, Jg. 30, Nr. 1, S. 15–22, 2020.

[194] C. M. Harmonosky und S. F. Robohn, „Real-time scheduling in computer integrated manufacturing: a review of recent research,“ *International Journal of Computer Integrated Manufacturing*, Jg. 4, Nr. 6, S. 331–340, 1991.

[195] T. Gabel und M. Riedmiller, „Adaptive Reactive Job-Shop Scheduling with Reinforcement Learning Agents,“ *International Journal of Information Technology and Intelligent Computing*, Jg. 24, Nr. 4, S. 14–18, 2008.

[196] E.-G. Talbi, *Metaheuristics: From design to implementation*. Hoboken: Wiley, 2009.

[197] T. Chaari, S. Chaabane, N. Aissani und D. Trentesaux, „Scheduling under uncertainty: Survey and research directions,“ in *2014 International Conference on Advanced Logistics and Transport (ICALT)*, IEEE, 2014, S. 229–234.

[198] I. Sabuncuoglu und M. Bayız, „Analysis of reactive scheduling problems in a job shop environment,“ *European Journal of Operational Research*, Jg. 126, Nr. 3, S. 567–586, 2000.

[199] G. E. Vieira, J. W. Herrmann und E. Lin, „Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods,“ *Journal of Scheduling*, Jg. 6, Nr. 1, S. 39–62, 2003.

[200] I. Sabuncuoglu und S. Goren, „Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research," *International Journal of Computer Integrated Manufacturing*, Jg. 22, Nr. 2, S. 138–157, 2009.

[201] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason und O. Rose, „A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations," *Journal of Scheduling*, Jg. 14, Nr. 6, S. 583–599, 2011.

[202] I. R. Uhlmann und E. M. Frazzon, „Production rescheduling review: Opportunities for industrial integration and practical applications," *Journal of Manufacturing Systems*, Jg. 49, S. 186–193, 2018.

[203] L. Nie, L. Gao, P. Li und X. Li, „A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates," *Journal of Intelligent Manufacturing*, Jg. 24, Nr. 4, S. 763–774, 2013.

[204] J. Heger, „Dynamische Regelselektion in der Reihenfolgeplanung," Dissertation, Universität Bremen, 2014.

[205] G. Chryssolouris und V. Subramaniam, „Dynamic scheduling of manufacturing job shops using genetic algorithms," *Journal of Intelligent Manufacturing*, Jg. 12, Nr. 3, S. 281–293, 2001.

[206] T. van Brackel, „Adaptive Steuerung flexibler Werkstattfertigungssysteme: Nutzung moderner Informations- und Kommunikationstechnologien zur effizienten Produktionssteuerung unter Echtzeitbedingungen," Dissertation, Universität Paderborn, 2009.

[207] M. Niehues, F. Buschle und G. Reinhart, „Adaptive Job-shop Control Based on Permanent Order Sequencing," *Procedia CIRP*, Jg. 33, S. 127–132, 2015.

[208] M. Niehues, „Adaptive Produktionssteuerung für Werkstattfertigungssysteme durch fertigungsbegleitende Reihenfolgebildung," Dissertation, Technische Universität München, 2016.

[209] M. Niehues, M. Blum, U. Teschemacher und G. Reinhart, „Adaptive job shop control based on permanent order sequencing," *Production Engineering*, Jg. 12, Nr. 1, S. 65–71, 2018.

[210] Z. Cao, L. Zhou, B. Hu und C. Lin, „An Adaptive Scheduling Algorithm for Dynamic Jobs for Dealing with the Flexible Job Shop Scheduling Problem," *Business & Information Systems Engineering*, Jg. 61, Nr. 3, S. 299–309, 2019.

[211] E. G. Birgin, P. Feofiloff, C. G. Fernandes, E. L. de Melo, M. T. I. Oshiro und D. P. Ronconi, „A MILP model for an extended version of the Flexible Job Shop Problem," *Optimization Letters*, Jg. 8, Nr. 4, S. 1417–1431, 2014.

[212] L. Wang, S. Wang und X. Zheng, „A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times,“ *Journal of Automatica Sinica*, Jg. 3, Nr. 3, S. 235–246, 2016.

[213] N. Al-Hinai und T. Y. ElMekkawy, „Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm,“ *International Journal of Production Economics*, Jg. 132, Nr. 2, S. 279–291, 2011.

[214] J. Fang und Y. Xi, „A rolling horizon job shop rescheduling strategy in the dynamic environment,“ *The International Journal of Advanced Manufacturing Technology*, Jg. 13, Nr. 3, S. 227–232, 1997.

[215] I. Sabuncuoglu und S. Karabuk, „Rescheduling frequency in an FMS with uncertain processing times and unreliable machines,“ *Journal of Manufacturing Systems*, Jg. 18, Nr. 4, S. 268–283, 1999.

[216] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, [Nachdr.], Ser. A Bradford book. Cambridge, Mass.: MIT Press, 1975.

[217] S. Koch, *Genetische Algorithmen für das Order Batching-Problem in manuellen Kommissioniersystemen*, Ser. Produktion und Logistik. Wiesbaden: Springer Gabler, 2014.

[218] J. M. Keller, D. Liu und D. B. Fogel, *Fundamentals of computational intelligence: Neural networks, fuzzy systems, and evolutionalry computation*, Ser. IEEE Press series on computational intelligence. Hoboken NJ und Piscataway NJ: Wiley und IEEE Press, 2016.

[219] S. N. Sivanandam und S. N. Deepa, *Introduction to genetic algorithms*. Berlin, New York: Springer, 2008.

[220] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, G. Ruß und M. Steinbrecher, *Computational Intelligence*. Wiesbaden: Vieweg+Teubner Verlag, 2011.

[221] V. Nissen, *Einführung in Evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution*, Ser. Computational Intelligence. Wiesbaden: Vieweg+Teubner Verlag, 1997.

[222] M. Kumar, M. Husain, N. Upreti und D. Gupta, „Genetic Algorithm: Review and Application,“ *SSRN Electronic Journal*, 2010.

[223] Z. Michalewicz und D. B. Fogel, *How to Solve It: Modern Heuristics*, Second, revised and Extended edition. Berlin, Heidelberg: Springer, 2004.

[224] C. R. Reeves, „Genetic Algorithms,“ in *Handbook of Metaheuristics*, Ser. International Series in Operations Research & Management Science, M. Gendreau und J.-Y. Potvin, Hrsg., Bd. 146, Boston, MA: Springer US, 2010, S. 109–139.

[225] J. Pätz, *Soft Computing in der Bioinformatik: Eine grundlegende Einführung und Übersicht*, Ser. EXamen.press. Berlin, Heidelberg: Springer, 2006.

[226] T.-P. Hong, H.-S. Wang, W.-Y. Lin und W.-Y. Lee, „Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process," *Applied Intelligence*, Jg. 16, Nr. 1, S. 7–17, 2002.

[227] A. J. Umbarkar und P. D. Sheth, „Crossover operators in genetic algorithms: a review," *ICTACT Journal on Soft Computing*, Jg. 06, Nr. 01, S. 1083–1092, 2015.

[228] D. H. Wolpert und W. G. Macready, „No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, Jg. 1, Nr. 1, S. 67–82, 1997.

[229] M. Seebode und W. Gerth, „Echtzeitsystem für einen zweibeinigen Roboter mit adaptiver Bahnplanung," in *Mobilität und Echtzeit*, Ser. Informatik aktuell, W. Brauer, P. Holleczek und B. Vogel-Heuser, Hrsg., Berlin, Heidelberg: Springer, 2008, S. 88–97.

[230] K.-M. Lee, T. Yamakawa und K.-M. Lee, „A Genetic Algorithm for General Machine Scheduling Problems," *International Journal of Knowledge-Based Electronic*, Jg. 2, 1998.

[231] X. Shao, X. Li, L. Gao und C. Zhang, „Integration of process planning and scheduling - A modified genetic algorithm-based approach," *Computers & Operations Research*, Jg. 36, Nr. 6, S. 2082–2096, 2009.

[232] W. Struckmann und D. Wätjen, *Mathematik für Informatiker: Grundlagen und Anwendungen*, 2. Aufl. 2016. Berlin, Heidelberg: Springer und Imprint: Springer Vieweg, 2016.

[233] C. Büsing, *Graphen- und Netzwerkoptimierung*. Heidelberg: Spektrum Akademischer Verlag, 2010.

[234] P. Hartmann, *Mathematik für Informatiker: Ein praxisbezogenes Lehrbuch*, 5. Aufl. 2012. Wiesbaden: Vieweg+Teubner Verlag, 2012.

[235] F. Pezzella, G. Morganti und G. Ciaschetti, „A genetic algorithm for the Flexible Job-shop Scheduling Problem," *Computers and Operations Research*, Jg. 35, Nr. 10, S. 3202–3212, 2008.

[236] G. Zhang, L. Gao und Y. Shi, „An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, Jg. 38, Nr. 4, S. 3563–3573, 2011.

[237] J. Zhang, G. Ding, Y. Zou, S. Qin und J. Fu, „Review of job shop scheduling research and its new perspectives under Industry 4.0," *Journal of Intelligent Manufacturing*, Jg. 30, Nr. 4, S. 1809–1830, 2019.

[238] A. Boccalatte, A. Gozzi, M. Paolucci, V. Queirolo und M. Tamoglia, „A multi-agent system for dynamic just-in-time manufacturing production scheduling,“ in *2004 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, 2004, S. 5548–5553.

[239] Y. Wei, K. Gu, H. Liu und D. Li, „Contract Net based Scheduling Approach using Interactive Bidding for Dynamic Job Shop Scheduling,“ in *2007 IEEE International Conference on Integration Technology*, IEEE, 2007, S. 281–286.

[240] W. Shen und D. H. Norrie, „Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey,“ *Knowledge and Information Systems*, Jg. 1, Nr. 2, S. 129–156, 1999.

[241] S. Bussmann und K. Schild, „Self-organizing manufacturing control: an industrial application of agent technology,“ in *Proceedings of the 4th International Conference on MultiAgent Systems*, IEEE, 2000, S. 87–94.

[242] S. Bussmann und K. Schild, „An agent-based approach to the control of flexible production systems,“ in *Proceedings of the 8th International Conference on Emerging Technologies and Factory Automation*, IEEE, 2001, S. 481–488.

[243] K. Schild und S. Bussmann, „Self-organization in manufacturing operations,“ *Communications of the ACM*, Jg. 50, Nr. 12, S. 74–79, 2007.

[244] A. Rajabinasab und S. Mansour, „Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach,“ *The International Journal of Advanced Manufacturing Technology*, Jg. 54, Nr. 9-12, S. 1091–1107, 2011.

[245] I. Badr, „Agent-based dynamic scheduling for flexible manufacturing systems,“ Dissertation, Universität Stuttgart, 2011.

[246] S. Wang, J. Wan, Di Li und C. Zhang, „Implementing Smart Factory of Industrie 4.0: An Outlook,“ *International Journal of Distributed Sensor Networks*, Jg. 12, Nr. 1, S. 3 159 805, 2016.

[247] S. Wang, C. Zhang, C. Liu, Di Li und H. Tang, „Cloud-assisted interaction and negotiation of industrial robots for the smart factory,“ *Computers & Electrical Engineering*, Jg. 63, S. 66–78, 2017.

[248] M. Leusin, E. Frazzon, M. Uriona Maldonado, M. Kück und M. Freitag, „Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era,“ *Technologies*, Jg. 6, Nr. 4, S. 107, 2018.

[249] M. E. Leusin, M. Kück, E. M. Frazzon, M. U. Maldonado und M. Freitag, „Potential of a Multi-Agent System Approach for Production Control in Smart Factories,“ *IFAC-PapersOnLine*, Jg. 51, Nr. 11, S. 1459–1464, 2018.

[250] F. Gehlhoff und A. Fay, „On agent-based decentralized and integrated scheduling for small-scale manufacturing,“ *at - Automatisierungstechnik*, Jg. 68, Nr. 1, S. 15–31, 2020.

[251] F. Gehlhoff, A. Fay und B. Köster, „Dezentrales Scheduling für dynamische Systeme,“ *atp magazin*, Jg. 62, Nr. 9, S. 54–61, 2020.

[252] I. Kovalenko, D. Ryashentseva, B. Vogel-Heuser, D. Tilbury und K. Barton, „Dynamic Resource Task Negotiation to Enable Product Agent Exploration in Multi-Agent Manufacturing Systems,“ *IEEE Robotics and Automation Letters*, Jg. 4, Nr. 3, S. 2854–2861, 2019.

[253] I. Kovalenko, D. Tilbury und K. Barton, „The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems,“ *Control Engineering Practice*, Jg. 86, S. 105–117, 2019.

[254] I. Kovalenko, K. Barton und D. Tilbury, „Design and implementation of an intelligent product agent architecture in manufacturing systems,“ in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2017, S. 1–8.

[255] S. Rehberger, L. Spreiter und B. Vogel-Heuser, „An agent approach to flexible automated production systems based on discrete and continuous reasoning,“ in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, IEEE, 2016, S. 1249–1256.

[256] I. Kovalenko, „Intelligent Product Agents for Multi-Agent Control of Manufacturing Systems,“ Dissertation, University of Michigan, 2020.

[257] F. Gehlhoff, L. Wiegandt und A. Fay, „Agent-Based Decentralised Process Planning and Evolutionary Change Propagation,“ in *IEEE ICIT*, 2021.

[258] F. Gehlhoff und A. Fay, „Agent-based decentralised architecture for multi-stage and integrated scheduling,“ in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2020, S. 1443–1446.

[259] R. Erol, C. Sahin, A. Baykasoglu und V. Kaplanoglu, „A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems,“ *Applied Soft Computing*, Jg. 12, Nr. 6, S. 1720–1732, 2012.

[260] P. Leitão, F. Casais und F. Restivo, „Holonic Manufacturing Control: A Practical Implementation,“ in *Emerging Solutions for Future Manufacturing Systems*, Ser. IFIP International Federation for Information Processing, L. M. Camarinha-Matos, Hrsg., Bd. 159, Boston: Kluwer Academic Publishers, 2005, S. 33–44.

[261] P. Leitão, „An agile and adaptive holonic architecture for manufacturing,“ Dissertation, Universität Porto, 2004.

[262] C. Sahin, M. Demirtas, R. Erol, A. Baykasoğlu und V. Kaplanoğlu, „A multi-agent based approach to dynamic scheduling with flexible processing capabilities,“ *Journal of Intelligent Manufacturing*, Jg. 28, Nr. 8, S. 1827–1845, 2017.

[263] S. Fatima, M. Wooldridge und S. Kraus, *Principles of automated negotiation*. Cambridge: Cambridge University Press, 2015.

[264] The Foundation for Intelligent Physical Agents, *FIPA Contract Net Interaction Protocol Specification*, 2002.

[265] T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*. Cambridge: Cambridge University Press, 2016.

[266] V. Krishna, *Auction theory*. San Diego: Academic Press, 2002.

[267] M. Aydin und E. Öztemel, „Dynamic job-shop scheduling using reinforcement learning agents,“ *Robotics and Autonomous Systems*, Jg. 33, Nr. 2-3, S. 169–178, 2000.

[268] D. Creighton und S. Nahavandi, „The application of a reinforcement learning agent to a multi-product manufacturing facility,“ in *2002 IEEE International Conference on Industrial Technology, 2002. IEEE ICIT '02.*, Bd. 2, 2002, 1229–1234 vol.2.

[269] Y.-R. Shiue, K.-C. Lee und C.-T. Su, „Real-time scheduling for a smart factory using a reinforcement learning approach,“ *Computers & Industrial Engineering*, Jg. 125, S. 604–614, 2018.

[270] Y.-F. Wang, „Adaptive job shop scheduling strategy based on weighted Q-learning algorithm,“ *Journal of Intelligent Manufacturing*, Jg. 31, Nr. 2, S. 417–432, 2020.

[271] T. Zhang, „Material Flow Control on Shop Floors in Job Shops: Release, Routing, and Sequencing,“ Dissertation, Universität der Bundeswehr München, 2018.

[272] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness und M. G. Bellemare et al., „Human-level control through deep reinforcement learning,“ *Nature*, Jg. 518, Nr. 7540, S. 529–533, 2015.

[273] J. Schulman, S. Levine, P. Moritz, M. I. Jordan und Abbeel Pieter, „Trust Region Policy Optimization,“ *arXiv preprint arXiv:1502.05477*, 2015.

[274] J. Schulman, F. Wolski, P. Dhariwal, A. Radford und O. Klimov, „Proximal policy optimization algorithms,“ *arXiv preprint arXiv:1707.06347*, 2017.

[275] J. P. Usuga Cadavid, S. Lamouri, B. Grabot, R. Pellerin und A. Fortin, „Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0,“ *Journal of Intelligent Manufacturing*, Jg. 31, Nr. 6, S. 1531–1558, 2020.

[276] A. Kuhnle, „Adaptive Order Dispatching based on Reinforcement Learning: Application in a Complex Job Shop in the Semiconductor Industry,“ Dissertation, Karlsruher Institut für Technologie, 2020.

[277] B. Waschneck, „Autonome Entscheidungsfindung in der Produktionssteuerung komplexer Werkstattfertigungen,“ Dissertation, Universität Stuttgart, 2020.

[278] S. Mahadevan und G. Theocharous, „Optimizing Production Manufacturing Using Reinforcement Learning,“ *FLAIRS Conference*, Nr. 372, 1998.

[279] S. Riedmiller und M. Riedmiller, „A neural reinforcement learning approach to learn local dispatching policies in production scheduling,“ *IJCAI'99: Proceedings of the 16th international joint conference on Artificial intelligence*, Nr. 2, S. 764–769, 1999.

[280] T. Altenmüller, T. Stüker, B. Waschneck, A. Kuhnle und G. Lanza, „Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints,“ *Production Engineering*, Jg. 14, Nr. 3, S. 319–328, 2020.

[281] A. Kuhnle, N. Röhrig und G. Lanza, „Autonomous order dispatching in the semiconductor industry using reinforcement learning,“ *Procedia CIRP*, Jg. 79, S. 391–396, 2019.

[282] A. Kuhnle, L. Schäfer, N. Stricker und G. Lanza, „Design, Implementation and Evaluation of Reinforcement Learning for an Adaptive Order Dispatching in Job Shop Manufacturing Systems,“ *Procedia CIRP*, Jg. 81, S. 234–239, 2019.

[283] A. Kuhnle, J.-P. Kaiser, F. Theiß, N. Stricker und G. Lanza, „Designing an adaptive production control system using reinforcement learning,“ *Journal of Intelligent Manufacturing*, Jg. 32, Nr. 3, S. 855–876, 2021.

[284] M.-A. Dittrich und S. Fohlmeister, „Cooperative multi-agent system for production control using reinforcement learning,“ *CIRP Annals*, Jg. 69, Nr. 1, S. 389–392, 2020.

[285] T. Gabel und M. Riedmiller, „Scaling Adaptive Agent-Based Reactive Job-Shop Scheduling to Large-Scale Problems,“ in *2007 IEEE Symposium on Computational Intelligence in Scheduling*, IEEE, 2007, S. 259–266.

[286] S. Qu, J. Wang, S. Govil und J. O. Leckie, „Optimized Adaptive Scheduling of a Manufacturing Process System with Multi-skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach,“ *Procedia CIRP*, Jg. 57, S. 55–60, 2016.

[287] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmuller, T. Bauernhansl, A. Knapp und A. Kyek, „Deep reinforcement learning for semiconductor production scheduling,“ in *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, IEEE, 2018, S. 301–306.

[288] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp und A. Kyek, „Optimization of global production scheduling with deep reinforcement learning,“ *Procedia CIRP*, Jg. 72, S. 1264–1269, 2018.

[289] S. Bär, D. Turner, P. K. Mohanty, V. Samsonov, J. R. Bakakeu und T. Meisen, „Multi Agent Deep Q-Network Approach for Online Job Shop Scheduling in Flexible Manufacturing," in *International Conference on Manufacturing System and Multiple Machines*, World Academy of Science, Engineering and Technology (WASET), 2020.

[290] M. C. May, L. Kiefer, A. Kuhnle, N. Stricker und G. Lanza, „Decentralized Multi-Agent Production Control through Economic Model Bidding for Matrix Production Systems," *Procedia CIRP*, Jg. 96, S. 3–8, 2021.

[291] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham: Springer International Publishing, 2018.

[292] D. Schröder, *Intelligente Verfahren*. Berlin, Heidelberg: Springer, 2010.

[293] C. J. C. H. Watkins und P. Dayan, „Q-learning," *Machine Learning*, Jg. 8, Nr. 3-4, S. 279–292, 1992.

[294] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare und J. Pineau, „An Introduction to Deep Reinforcement Learning," *Foundations and Trends in Machine Learning*, Jg. 11, Nr. 3-4, S. 219–354, 2018.

[295] O. Nachum, M. Norouzi, K. Xu und D. Schuurmans, „Bridging the Gap Between Value and Policy Based Reinforcement Learning," *arXiv preprint arXiv:1702.08892*, 2017.

[296] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner und S. Levine, „Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic," *arXiv preprint arXiv:1611.02247*, 2016.

[297] J. Schulman, P. Moritz, S. Levine, M. I. Jordan und Abbeel Pieter, „High-Dimensional Continuous Control Using Generalized Advantage Estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[298] R. J. Williams, „Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, Jg. 8, Nr. 3-4, S. 229–256, 1992.

[299] R. S. Sutton und A. Barto, *Reinforcement learning: An introduction*, Ser. Adaptive computation and machine learning. Cambridge Massachusetts: The MIT Press, 1998.

[300] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver und K. Kavukcuoglu, „Asynchronous Methods for Deep Reinforcement Learning," *Proceedings of Machine Learning Research*, S. 1928–1937, 2016.

[301] C. E. Shannon, „A Mathematical Theory of Communication," *Bell System Technical Journal*, Jg. 27, Nr. 3, S. 379–423, 1948.

[302] S. Huang und S. Ontanon, „A Closer Look at Invalid Action Masking in Policy Gradient Algorithms," *arXiv preprint arXiv:2006.14171*, 2020.

[303] J. Liu, P. Hou, L. Mu, Y. Yu und C. Huang, „Elements of Effective Deep Reinforcement Learning towards Tactical Driving Decision Making," *arXiv preprint arXiv:1802.00332*, 2018.

[304] D. P. Kingma und J. Ba, „Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[305] S. Ruder, „An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[306] G. van Rossum, „Python reference manual," *Department of Computer Science*, Nr. R 9525, 1995.

[307] J. D. Hunter, „Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, Jg. 9, Nr. 3, S. 90–95, 2007.

[308] M. L. Waskom, „seaborn: statistical data visualization," *Journal of Open Source Software*, Jg. 6, Nr. 60, S. 3021, 2021.

[309] S. Bangsow, *Tecnomatix Plant Simulation: Modeling and Programming by Means of Examples*. Cham: Springer International Publishing, Imprint und Springer, 2016.

[310] M. Lekic und G. Gardasevic, „IoT sensor integration to Node-RED platform," in *2018 17th International Symposium Infoteh-Jahorina (Infoteh)*, IEEE, 2018, S. 1–5.

[311] S. Sicari, A. Rizzardi und A. Coen-Porisini, „Smart transport and logistics: A Node-RED implementation," *Internet Technology Letters*, Jg. 2, Nr. 2, 2019.

[312] F.-A. Fortin, F.-M. de Rainville, M.-A. Gardner, M. Parizeau und C. Gagné, „DEAP: Evolutionary Algorithms Made Easy," *Journal of Machine Learning Research*, S. 2171–2175, 2012.

[313] D. Merkel, „Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, Jg. 2014, Nr. 239, 2014.

[314] R. A. Light, „Mosquitto: server and client implementation of the MQTT protocol," *The Journal of Open Source Software*, Jg. 2, Nr. 13, S. 265, 2017.

[315] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury und G. Chanan et al., „PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv preprint arXiv:1912.01703*, 2019.

[316] S. Wenzel, M. Weiß, S. Collisi-Böhmer, H. Pitsch und O. Rose, *Qualitätskriterien für die Simulation in Produktion und Logistik*. Berlin, Heidelberg: Springer, 2008.

[317] M. Rabe, S. Spieckermann und S. Wenzel, *Verifikation und Validierung für die Simulation in Produktion und Logistik*. Berlin, Heidelberg: Springer, 2008.

[318] H.-J. Bungartz, S. Zimmer, M. Buchholz und D. Pflüger, *Modellbildung und Simulation*. Berlin, Heidelberg: Springer, 2013.

[319] J. Heaton, *Introduction to neural networks with Java*, 2. ed., 1. print. St. Louis: Heaton Research, 2008.