Universität der Bundeswehr München

Automatisierte Auswahl von Algorithmen für das dynamische Fahrzeugwegeplanungsproblem

Dipl.-Inf. Thomas Mayer

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Gutachter/Gutachterin:
Univ.-Prof. Dr. Oliver Rose
Univ.-Prof. Dr.-Ing. Markus Siegle

Die Dissertation wurde am 05. August 2019 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 07. November 2019 angenommen. Die mündliche Prüfung fand am 22. November 2019 statt.

Prüfungskommission:

Vorsitzender:	Univ.-Prof. Dr. Cornelius Greither
1. Gutachter:	Univ.-Prof. Dr. Oliver Rose
2. Gutachter:	Univ.-Prof. Dr.-Ing. Markus Siegle
1. Prüfer:	Univ.-Prof. Dr. Gunnar Teege
2. Prüfer:	Univ.-Prof. Dr. Michael Koch
3. Prüfer:	Univ.-Prof. Dr.-Ing. Andreas Karcher

Zusammenfassung

Um die zunehmenden Bedürfnisse unserer modernen Gesellschaft zu befriedigen, müssen immer komplexere Routingprobleme gelöst werden. Allein in den letzten 20 Jahren hat sich die Anzahl der zu befördernden Sendungen in Deutschland um fast 100% erhöht. Doch nicht nur im transportierenden Gewerbe, sondern auch in der Intralogistik, im Dienstleistungssektor oder im Nahverkehr steigen die Anforderungen, auch an die Flexibilität der Lösungen, stetig. Die zunehmende Vernetzung erlaubt das dynamische Anpassen von bereits geplanten Touren. So können Kundenanfragen online in bestehende Routen integriert werden, um so effizient wie möglich auf neue Nachfragen zu reagieren. Aufgrund des gesteigerten Interesses an der dynamischen Fahrzeugwegeplanung (engl. *Dynamic Vehicle Routing Problem* (DVRP)) existieren eine Vielzahl von unterschiedlichen Lösungsansätzen und stetig werden neue entwickelt. Einem Anwender stellt sich aus diesem Grund zunehmend die Frage nach dem besten Ansatz für eine konkrete Problemstellung. Die vorliegende Arbeit widmet sich dieser Fragestellung und wirft dabei, unter anderem, die folgenden Fragen auf. Welche Problemstellungen sind für unterschiedliche Algorithmen besonders schwer bzw. leicht zu lösen? Wie beschreibt man Beziehungen zwischen Lösungsansätzen und Probleminstanzen? Anhand welcher Charakteristika können Instanzen voneinander unterschieden werden? Für die Beantwortung dieser Fragestellungen werden Problemeigenschaften für das DVRP entwickelt, die es erstmalig ermöglichen Probleminstanzen anhand ihrer Dynamik voneinander zu unterscheiden und Algorithmen fair miteinander zu vergleichen. Damit leistet die vorliegende Arbeit einen wesentlichen wissenschaftlichen Beitrag bei der Charakterisierung der Dynamik von Routing Problemstellungen. Mit einem umfangreichen Data-Farming-Experiment werden Erfahrungen über den Zusammenhang zwischen Lösungsqualität und Beschaffenheit von Probleminstanzen generiert. Das Data-Farming-Experiment basiert auf dem in dieser Arbeit eingeführten, quelloffenen Simulator für DVRP. Die gesammelten Erfahrungen werden mit Methoden der Data-Science untersucht und Wissen für die manuelle Auswahl von Algorithmen abgeleitet. Die Datenbasis und die erarbeiteten Eigenschaften bilden die Grundlage für die Konstruktion eines Algorithmus, der Probleminstanzen mit besonderen Ausprägungen bestimmter Problemeigenschaften sehr erfolgreich löst. Zusätzlich werden aus den generierten Erfahrungen Modelle mit Methoden des maschinellen Lernens abgeleitet, die einem Optimierer die automatisierte Auswahl von geeigneten Algorithmen für unbekannte Problemstellungen ermöglicht. Damit bereitet die vorliegende Arbeit die Grundlagen für die Beantwortung der Frage nach dem besten Lösungsansatz für eine dynamische Routingproblemstellung. Zusätzlich wird wesentlich zum Verständnis über die Beziehungen zwischen Problemdynamik und Verhalten von Algorithmen beigetragen.

Dynamisches Vehicle Routing, Simulation, Selektion von Algorithmen, maschinelles Lernen, künstliche neuronale Netze, Data-Farming, Data-Science

Abstract

To satisfy the needs of our modern society, more complex routing problems need to be resolved. In Germany, for example, within the last 20 years the number of transported consignments has increased by nearly 100%. This trend is not limited to the transportation sector. Within the service industry, in public transport or in intra logistics, more factors need to be considered. Especially the flexibility of created solutions is gaining in interest. The increased degree of interconnectedness allows the adaption of already planned tours. Customer requests are integrated online into existing routes, to respond efficiently to new demands. Due to the heightened interest in the *Dynamic Vehicle Routing Problem* (DVRP), varied new solution approaches are constantly developed. Hence, the user is confronted with the question of which the most fitting approach to the problem instance is. The presented thesis addresses this and also additional questions associated. Which problem instances are hard or easy to solve for algorithms? How do you describe the relationship between solution approach and problem instance? Which characteristics better distinguish instances? To answer these questions, problem features for the DVRP are developed. With the help of these features, it is now possible to distinguish problem instances based on their dynamism and to compare algorithms in a fair way. This thesis therefore provides a significant scientific contribution to the characterization of the dynamism of routing problem instances. With an extensive data farming experiment, observations about the relation between solution quality and character of a problem are generated. The experiment bases on the open-source simulator for DVRP, introduced in this thesis. The aggregated observations are evaluated with methods of data science. Consequently, knowledge for the manual selection of algorithms is derived. The generated data basis and the introduced problem features are used to construct an algorithm which solves problem instances with special features effectively. Additionally, models are derived from the collective observations with the help of machine learning methods. The constructed models are used to implement an optimizer which can successfully select algorithms for unknown problem instances. Therefore, the presented thesis lays the foundation in answering the question to the best solution approach for an instances of a dynamic routing problem.

dynamic vehicle routing, simulation, algorithm selection, machine learning, artificial neural networks, data farming, data science

Danksagung

Besonders danken möchte ich meinem Doktorvater Herrn Univ.-Prof. Dr. Oliver Rose für die Möglichkeit der Promotion, für das mir entgegengebrachte Zutrauen und Vertrauen und für den wissenschaftlichen und methodischen Rat bei der Bearbeitung des Themengebietes. Ein großer Dank gilt ebenfalls Herrn Univ.-Prof. Dr.-Ing. Markus Siegle für die zahlreichen Anmerkungen, die zur Qualität der Arbeit beigetragen haben.

Meinem geschätzten Kollegen, Herrn Dr. Tobias Uhlig, danke ich vor allem für die intensive Unterstützung durch unzählige fachliche Diskussionen und Anmerkungen. Viele umgesetzte Ideen haben ihren Ursprung in diesen Gesprächen. Bedanken möchte ich mich auch bei allen Kollegen, die mich während meiner Promotion begleitet haben.

Ein besonders inniger Dank gilt meiner Frau Doreen Chang und meinen beiden wundervollen Kindern Henry und Elli. Immer wieder war und ist mir meine Familie ein wichtiger Motivator aber auch Ruhepol. Das gilt auch für meine Eltern, Beate und Wolfgang Mayer, ohne deren Unterstützung viel von dem Erreichten nicht möglich gewesen wäre.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Um die Bedürfnisse unserer modernen Gesellschaft zu befriedigen, müssen unzählige komplexe Routingprobleme gelöst werden. Ein signifikanter Anteil der Konsumgüter wird heutzutage über das Internet bestellt. Allein Amazon erwirtschaftete in Deutschland im Jahr 2018 einen Umsatz von 19,9 Mrd. US-Doller, eine Steigerung um 17,3% im Vergleich zum Jahr 2017 (Börsenblatt, 2019). Selbst Nahrungsmittelkonzerne liefern verderbliche Waren für den täglichen Bedarf bis vor die Haustür. Nach Esser und Kurte (2018) transportieren die Kurier-, Express- und Paketdienste in Deutschland im Jahr 2017 98% mehr Sendungen im Vergleich zum Jahr 2000. Ein enormer Zuwachs, der ausschließlich mit Unternehmenswachstum, zunehmender Digitalisierung und effizienten Algorithmen zu gewährleisten ist (Esser und Kurte, 2018). In das Zentrum der Aufmerksamkeit rücken zunehmend die dynamischen Varianten der Routingprobleme. Hier ändern sich Probleminformationen über die Zeit. Neue Anfragen an Transporte oder Dienstleistungen werden den Planern bekannt und können aufgrund von zunehmender Vernetzung und verbesserter Telemetrie und Telematik noch während der Durchführung in die aktuellen Routen integriert werden. Ein typisches Anwendungsbeispiel einer solchen dynamischen Problemstellung, die auch als *Dynamic Vehicle Routing Problem* (DVRP) bezeichnet wird, ist die Durchführung von Wartungen und die Reparatur von Anlagen oder Maschinen. Servicetechniker verlassen das Depot, um geplante Wartungen durchzuführen. Während der Durchführung werden Störungen anderer Anlagen oder Maschinen bekannt, die von den Technikern behoben werden müssen. Die geplanten Routen werden den neuen Gegebenheiten so effizient wie möglich angepasst. Auch moderne Varianten des Personentransports müssen sich den Herausforderungen dynamischer Routingprobleme stellen. Beförderungsaufträge werden dynamisch in bestehende Routen integriert, um den Bedürfnissen der Kunden effizient zu begegnen. Neben der Wichtigkeit von dynamischen Problemstellungen im Transport oder der Dienstleistungsbranche finden sich analoge Probleme auch in der Intralogistik. Werden bestellte Waren aus sehr großen Lagern zusammengetragen, muss bei neu eingehenden Bestellungen entschieden werden, wie diese so schnell wie möglich durch die Transportflotte zum Warenausgang transportiert werden können. Üblicherweise soll die Zeit zwischen Bestellvorgang und Auslieferung auf ein Minimum reduziert werden.

Die zunehmende Präsenz dynamischer Routingproblemstellungen zeigt sich ebenfalls im gesteigerten Interesses von wissenschaftlichen Institutionen. Psaraftis et al. (2016) beschreibt eine explosionsartige Steigerung der Veröffentlichungen seit dem Jahr 2000, die sich mit dem DVRP auseinandersetzen. Jedes Jahr werden verschiedene neue Ansätze zur Lösung für die unterschiedlichsten Variationen des DVRP vorgestellt. Eine von Pillac et al. (2013) vorgestellte Studie listet 154 Referenzen, die sich mit Lösungsansätzen für dynamische Problemstellungen auseinandersetzen (Psaraftis et al., 2016). Kontinuierlich werden neue Ansätze entwickelt. Bei der Vielzahl von Algorithmen und Ansätzen stellt sich zunehmend bei der tagtäglichen Bearbeitung dynamischer Probleme die Frage nach dem geeignetsten Lösungsansatz oder Algorithmus für eine konkrete Problemstellung. Diese Fragestellung ist die Hauptmotivation der vorliegenden Arbeit und soll anhand eines nachfolgenden Beispiels näher erläutert werden. Abbildung 1.1 zeigt zwei zufällig erzeugte Instanzen eines DVRP. Schwarze Punkte markieren Kunden, die einem Planer vor der Durchführung bekannt sind (statische Kunden). Dynamische Kunden, also solche, die erst während der Bearbeitung bekannt werden, sind mithilfe von Dreiecken visualisiert. Die Probleminstanzen sind sich relativ ähnlich. In beiden sind sieben statische und drei dynamische Kunden modelliert. Das Depot, also der Ausgangs- und Endpunkt einer Route, befindet sich im Bereich um das Zentrum der Instanz.

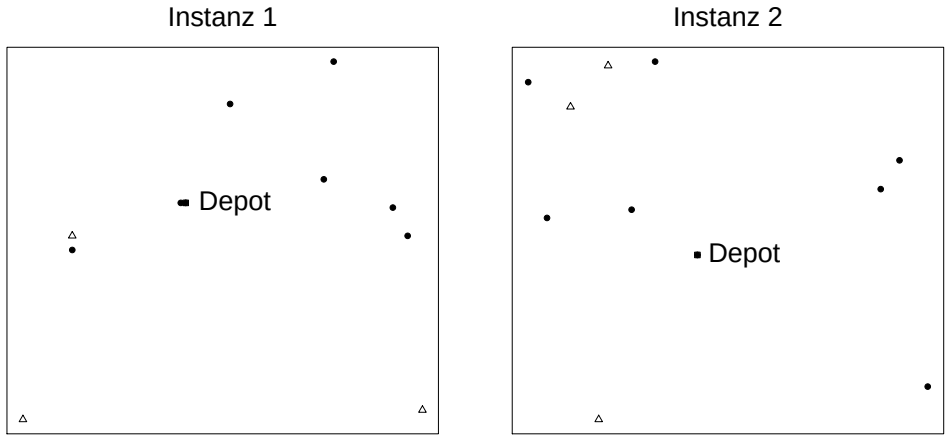

Abbildung 1.1: Zwei zufällig erzeugte DVRP-Instanzen (schwarze Punkte repräsentieren Kunden, die einem Planer bei der Erstellung einer initialen Lösung bekannt sind, Dreiecke visualisieren dynamische Kunden)

Abbildung 1.2 zeigt die beiden zufällig erzeugten Probleminstanzen, gelöst durch zwei verschiedene Lösungsansätze. In dem abgebildeten Zustand sind alle statischen und dynamischen Anfragen bedient und der Transporter bzw. Dienstleister befindet sich wieder am Ausgangspunkt. Die erste Erkenntnis, die aus den Abbildungen abgeleitet werden kann, ist, dass zwei verschiedene Lösungsansätze unterschiedliche Lösungen für eine Problemstellung generieren. Das muss nicht zwangsläufig so sein, ist aber auch nicht verwunderlich aufgrund der Erkenntnisse, abgeleitet aus den No-Free-Lunch-Theoremen, eingeführt von Wolpert und Macready (1997). Auf die No-Free-Lunch-Theoreme wird

im Folgenden noch genauer eingegangen.

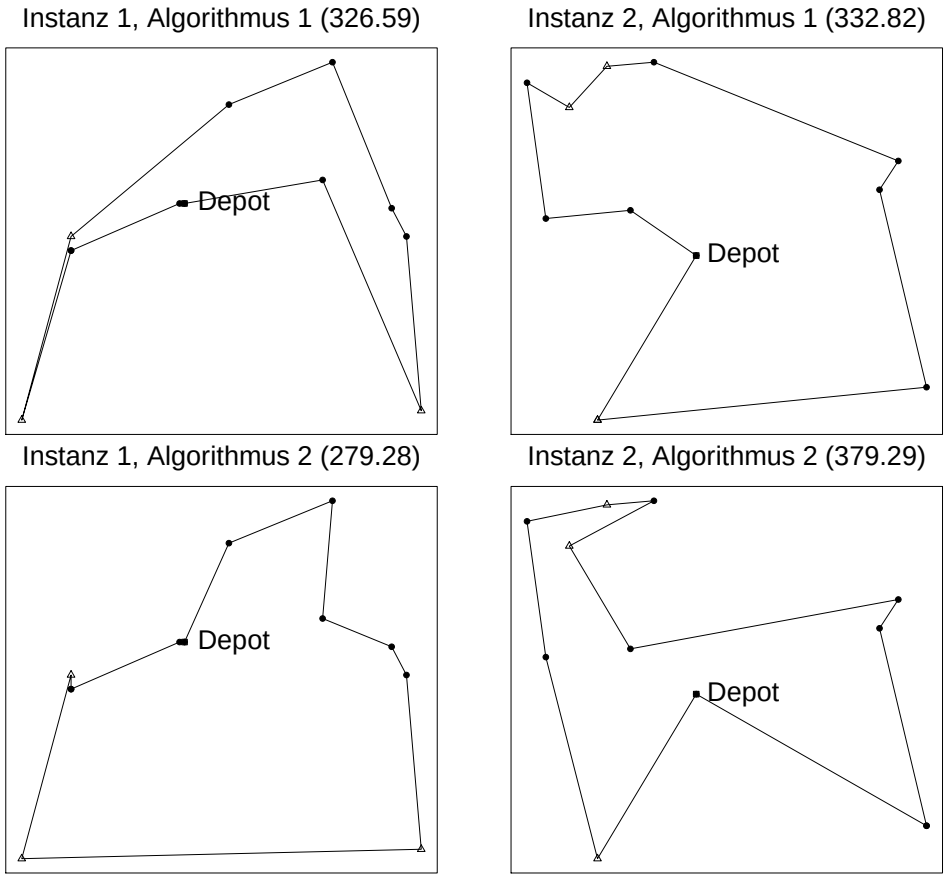

Abbildung 1.2: Die zufällig erzeugten DVRP-Instanzen 1 (erste Spalte) und 2 (zweite Spalte), gelöst durch die Algorithmen 1 und 2 (Linien visualisieren die durch den Algorithmus geplanten und abgefahrenen Routen)

Die zweite Erkenntnis, die aus der Abbildung 1.2 abgeleitet werden kann, ist, dass sich unterschiedliche Algorithmen für scheinbar ähnliche Probleminstanzen als die besseren herausstellen. Für Probleminstanz 1 erzeugt der Algorithmus 2 die beste Lösung. Probleminstanz 2 wird erfolgreicher durch Algorithmus 1 gelöst. Die Frage nach dem geeignetsten Algorithmus für eine konkrete Problemstellung ist also essenziell und wirft weitere Fragestellungen auf. Wie können Probleminstanzen unterschieden bzw. miteinander verglichen werden? Welche Beziehungen gibt es zwischen Algorithmen und Probleminstanzen? Ist ein fairer Vergleich von Lösungsansätzen möglich, ohne die zugrundeliegende Probleminstanzen vergleichen zu können? Können Algorithmen gezielt für Problemstellungen anhand von Eigenschaften ausgewählt werden? Ist eine Auswahl manuell und automatisiert möglich? Gerade im Kontext von Online-Problemen, wie dem DVRP, ist es notwendig Eigenschaften von Problemstellungen zu kennen, die sich auf den Erfolg von Lösungsansätzen auswirken. Für diese Eigenschaften können dann gezielt Prognosemodelle entwickelt werden, um geeignete Algorithmen auszuwählen. Die informierte Wahl eines Algorithmus für ein Online-Problem ist essenziell, da eine online

getroffene Entscheidung nicht rückgängig gemacht werden kann und sich langfristig auf den Erfolg des gesamten Routings auswirkt.

Die vorliegende Dissertation adressiert alle aufgeworfenen Fragen. Bevor aber die konkreten Zielsetzungen der Arbeit aus den Fragestellungen abgeleitet werden, soll die Arbeit im nachfolgenden Abschnitt in den wissenschaftlichen Kontext, auch über die Domäne der Fahrzeugwegeplanung (Vehicle Routing) hinaus, eingeordnet werden.

1.1 Wissenschaftlicher Kontext

Die vorliegende Arbeit beschäftigt sich mit der Frage nach dem geeignetsten Lösungsansatz für eine konkrete Problemstellung und mithilfe welcher Problemeigenschaften ein solcher Ansatz identifiziert werden kann. Es wird der Frage nachgegangen, über welche dynamischen Aspekte eines DVRP Prognosen erstellt werden müssen, um informiert Algorithmen auszuwählen. Für die Beantwortung dieser Fragestellungen müssen vorerst Eigenschaften identifiziert werden, die einen signifikanten Einfluss auf die Lösungsqualität von Algorithmen für das DVRP haben. Um diese wesentlichen Charakteristika zu ergründen, muss der Zusammenhang zwischen der Lösungsqualität von Algorithmen und den Problemeigenschaften erforscht werden. Auch die manuelle und automatisierte Auswahl von Algorithmen für Probleminstanzen basiert auf diesem Zusammenhang.

Optimierungsprobleme wie das DVRP sind im Allgemeinen gekennzeichnet durch eine Vielzahl valider Lösungen. Jeder Lösung kann ein Wert zugeordnet werden (Kosten, Zielfunktionswert), der eine Beurteilung der Qualität der Lösung erlaubt. Üblicherweise suchen Lösungsansätze aus allen validen Lösungen diejenige mit der höchsten Lösungsqualität (Grötschel et al., 2012). Ein sehr einfacher Lösungsansatz ist es für alle validen Lösungen die Qualität zu bestimmen und diejenige Lösung mit der höchsten Qualität auszuwählen. Dieses Vorgehen ist oftmals ungeeignet, da die Menge an möglichen validen Lösungen sehr groß werden kann. Bei der Betrachtung einer Beispielproblemstellung wird dies schnell ersichtlich. Das Beispiel beschreibt ein *Travelling Salesman Problem* (TSP), auch bekannt unter der Bezeichnung „Problem des Handlungsreisenden“. Bei dem TSP muss ein Reisender eine Reihenfolge für den Besuch verschiedener Städte so wählen, dass keine Stadt mehrfach besucht wird und die Gesamtreisestrecke minimal ist. Soll der Reisende nur 15 Städte besuchen und ist die Reisestrecke c_{AB} zwischen den Städten A und B gleich der Reisestrecke c_{BA} zwischen B und A, ergeben sich schon rund 43 Milliarden mögliche Rundreisen. Die Anzahl der Reisen ergibt sich aus der Anzahl m aller mögliche Permutationen der 14 Städte. Es werden nur 14 Städte betrachtet, da die Startstadt für alle Rundreisen identisch ist. Da $c_{AB} = c_{BA}$ gilt, müssen nur die Hälfte $\frac{m}{2}$ der Permutationen betrachtet werden. Bei 20 zu bereisenden Städten ergeben sich schon rund 60 Billiarden mögliche Rundreisen. Wird für die Erzeugung einer Rundreise und die Berechnung der Qualität nur eine Mikrosekunde benötigt, muss ein sequenzieller Lösungsansatz noch rund zweitausend Jahre rechnen, um für 20 Städte alle möglichen Rundreisen zu erzeugen und hinsichtlich ihrer Qualität zu bewertet. Anhand dieses Beispiels erkennt man leicht, dass eine Enumeration über alle validen Lösungen kein

geeigneter Lösungsansatz ist. Aus diesem Grund werden üblicherweise heuristische Lösungsansätze zum Lösen von Optimierungsproblemen in der Komplexitätsklasse NP implementiert (Russell und Norvig, 2012). Ein Optimierungsproblem liegt in NP, wenn nur ein nichtdeterministischer Algorithmus, der das Problem in Polynomialzeit löst, gefunden werden kann (Wegener, 2013). Heuristische Verfahren verfügen über keinen formalen Algorithmus, der zwangsläufig die Lösung mit der höchsten Qualität erzeugt. Diese informierten (heuristischen) Lösungsansätze nutzen neben der Definition des Problems zusätzliches problemspezifisches Wissen, um Lösungen effizienter zu finden als ein uninformierter Lösungsansatz (Russell und Norvig, 2012).

Ein einfacher heuristische Ansatz für die Lösung des TSP wäre es, zum Beispiel, von der aktuellen Stadt zu der jeweils nächsten zu fahren, die noch nicht besucht wurde. Gibt es keine nicht-besuchten Städte mehr, kehrt man zum Ausgangspunkt, dem Depot, zurück. Ein solcher Algorithmus findet sehr effizient Lösungen, auch für sehr große Instanzen eines TSP. Abbildung 1.3 visualisiert die durch diese einfache Nächste-Nachbar (NN)-Heuristik generierten Lösungen für zwei TSP-Instanzen.

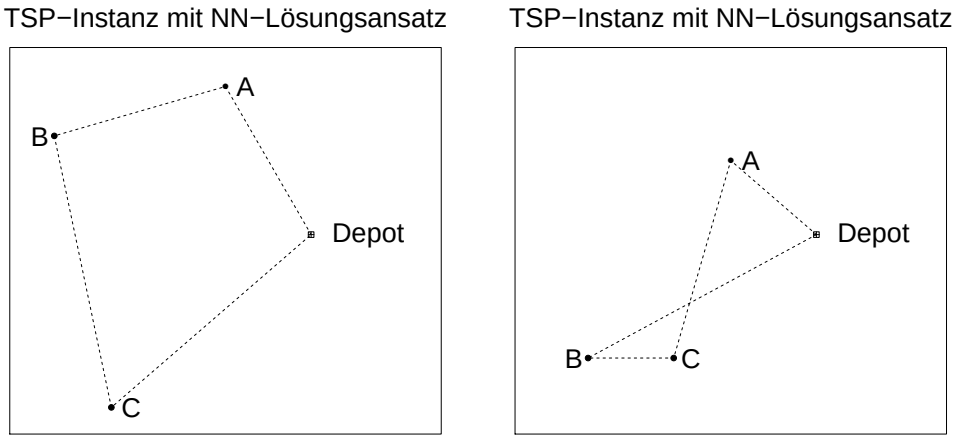

Abbildung 1.3: Visualisierung der Lösungen für zwei TSP-Instanzen konstruiert durch die NN-Heuristik

Für die in Abbildung 1.3 links darstellte Probleminstanz, findet die NN-Heuristik die optimale Rundreise. Keine andere Permutation der Städte A, B und C erzeugt eine Lösung mit größerer Qualität. Für die rechte Probleminstanz dargestellt in Abbildung 1.3 scheint die gefundene Lösung aufgrund der sich kreuzenden Wege hingegen nicht optimal zu sein. Es zeigt sich, dass $c_{AC} + c_{BDepot} > c_{AB} + c_{CDepot}$ und damit die Permutation (ABC) eine größere Lösungsqualität als die gefundene Lösung (ACB) hat. Die von der NN-Heuristik getroffene Entscheidung von Stadt A nach C zu fahren, weil $c_{AC} < c_{AB}$ gilt, erweist sich beim Betrachten der vollständigen Lösung als ungünstig. Es zeigt sich also, dass eine einfache Heuristik Lösungen mit hoher Qualität sehr effizient erzeugen kann. Es zeigt sich aber auch, dass die Qualität stark variieren kann und von der Beschaffenheit der Probleminstanz selbst abhängig ist. Die Frage, die sich an dieser Stelle stellt ist, ob ein heuristischer Lösungsansatz existiert, der für alle denkbaren

Probleminstanzen eine größere Lösungsqualität generiert als alle anderen existierenden heuristischen Ansätze. Diese aufkommende Fragestellung beantworteten Wolpert und Macready (1997) für alle kombinatorischen Optimierungsprobleme ganz klar mit Nein. Zusammengefasst zeigen Wolpert und Macready (1997), dass die durchschnittliche Lösungsqualität, erzeugt durch unterschiedliche heuristische Algorithmen, über alle möglichen Problemstellungen hinweg identisch ist. Der Nachweis wird mithilfe von zwei No-Free-Lunch-Theoremen sowohl für Offline-Probleme, als auch für Online-Probleme geführt. In einem Offline-Problem, wie, zum Beispiel, dem TSP, sind alle Probleminformationen a priori bekannt. Das in dieser Arbeit im Detail betrachtete DVRP ist ein Online-Problem, charakterisiert durch sich über die Zeit hinweg ändernde Probleminformationen. Für die Entwicklung und Analyse von Lösungsansätzen für Online-Probleme werden üblicherweise Simulationen implementiert (Grötschel et al., 2001). So können verschiedene Ansätze mithilfe von Experimenten mit einem Modell erprobt und nachvollzogen werden. Auf der Grundlage der No-Free-Lunch-Theoreme bestätigen Wolpert und Macready (1997), dass unterschiedliche heuristische Lösungsansätze ihre Daseinsberechtigung haben, weil diese unterschiedlichen Ansätze für unterschiedliche Problemstellungen konträre Lösungsqualitäten erzeugen. Implizit werfen Wolpert und Macready (1997) ebenfalls die Fragen auf, welche im Fokus der vorliegenden Arbeit stehen. Welche Lösungsansätze erzeugen für welche Gruppen von Problemstellungen die höchste Qualität, und wie kann man diese Gruppen voneinander unterscheiden? Welche Problemeigenschaften haben einen wesentlichen Einfluss auf die Lösungsqualität und können zum Trennen der Gruppen herangezogen werden? Für Offline-Problem können vor Beginn der Problemdurchführung beliebig viele Lösungsansätze evaluiert und anhand der erreichten Qualität ausgewählt werden. Bei einem Online-Problem hingegen werden die Auswirkungen, der durch einen Algorithmus getroffenen Entscheidungen, erst zur Laufzeit sichtbar und können nicht rückgängig gemacht werden. Aus diesem Grund ist gerade für Online-Optimierungsprobleme die Beantwortung der aufgeworfenen Fragestellungen essenziell. Die einfache NN-Heuristik, zum Beispiel, scheint TSP-Instanzen mit hoher Qualität lösen zu können, bei denen die Städte eher kreisförmig im Problemraum verteilt sind. Kommt es hingegen zu einer willkürlicheren Verteilung der Städte im Problemraum, versagt die NN-Heuristik. Demzufolge haben Optimierungsprobleme Eigenschaften, mithilfe derer die voraussichtliche Qualität einer Lösung durch einen heuristischen Lösungsansatz abgeleitet werden kann. Diese intrinsischen Eigenschaften einer Problemstellung werden in der vorliegenden Arbeit als Problemeigenschaften oder nur Eigenschaften bezeichnet. Abbildung 1.4 verbildlicht den Zusammenhang zwischen Problemstellung, Eigenschaften, Lösungsansätze und Lösungen. Optimierungsprobleme können mithilfe von Eigenschaften beschrieben werden. Lösungsansätze werden auf Problemstellungen angewendet und erzeugen Lösungen mit unterschiedlicher Qualität. Das Verständnis über den Zusammenhang zwischen Problemeigenschaften und Lösungsqualität kann dazu genutzt werden einen Optimierer zu implementieren. Dieser Optimierer soll anhand von Eigenschaften einen günstigen Lösungsansatz für eine Problemstellung empfehlen. Ein günstiger Lösungsansatz erzeugt

eine hohe Lösungsqualität für die Problemstellung. Ein einfacher Optimierer für das TSP-Beispiel aus Abbildung 1.3 empfiehlt als Lösungsansatz die einfach NN-Heuristik, wenn aus den Problemeigenschaften eine kreisförmige Anordnung der Städte ableitbar ist. Eine Alternative wird dann empfohlen, wenn diese kreisförmige Anordnung nicht erkennbar ist.

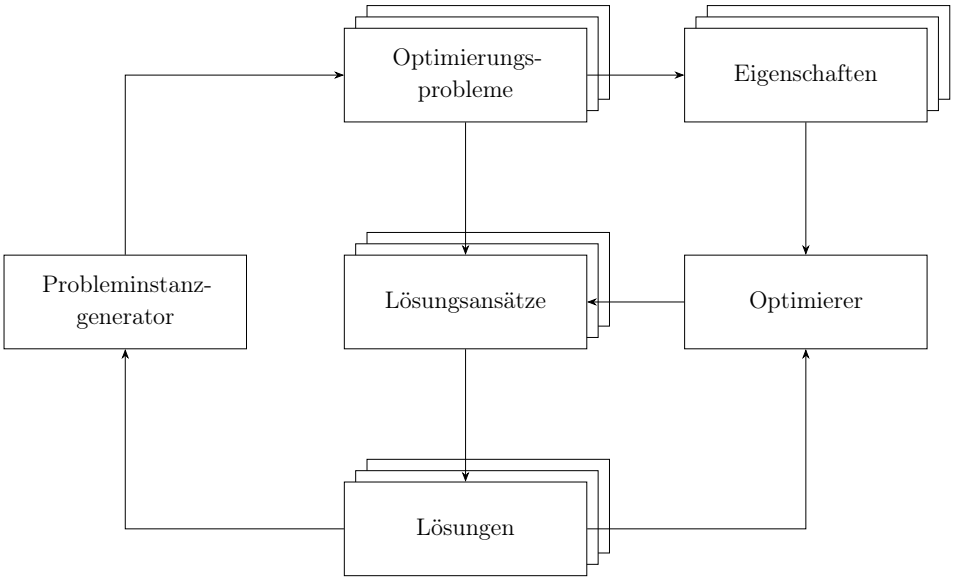

Abbildung 1.4: Übersicht über die wichtigsten Zusammenhänge zwischen Optimierungsproblem, Lösungsansatz und Algorithmenauswahl basierend auf Problemeigenschaften

Schon Rice (1976) diskutiert die Auswahl von Algorithmen für Problemstellungen basierend auf Problemeigenschaften und führt das *Algorithm Selection Problem* (ASP) und ein zugehöriges Framework ein. In frühen Arbeiten, die das ASP diskutieren, werden häufig Algorithmen gesucht, die eine Lösung besonders schnell konstruieren. In späteren Veröffentlichungen steht auch die Güte der konstruierten Lösung im Vordergrund. Das ASP wird ausführlich im weiteren Verlauf dieser Arbeit diskutiert. Abbildung 1.4 erweitert das von Rice (1976) eingeführte Framework um die Komponente Probleminstanzgenerator. Die Idee ist hier das Implementieren des Optimierers mit der zielgerichteten Generierung von Optimierungsprobleminstanzen zu unterstützen. Die Probleminstanzen sollen möglichst hohe bzw. niedrige Lösungsqualität für die Lösungsansätze aufweisen. In solchen Instanzen werden die Problemeigenschaften, die die Qualität der Lösung des Problems begünstigen bzw. verhindern besonders ausgeprägt sein. Die besonders starke Ausprägung wichtiger Problemeigenschaften unterstützt die Implementierung des Optimierers. Das zielgerichtete Erzeugen von Probleminstanzen ist im Kontext von Algorithmenauswahl ein übliches Vorgehen (van Hemert, 2006, Smith-Miles et al., 2010).

1.2 Wissenschaftlicher Beitrag und Zielsetzungen

In der Literatur werden nur sehr wenige Eigenschaften, die die Dynamik einer Probleminstanz charakterisieren, für das DVRP beschrieben. Für die Auswahl von Lösungsansätzen für eine konkrete Problemstellung ist eine genaue Beschreibung der Instanzen mithilfe von Eigenschaften notwendig. Ein Schwerpunkt der vorliegenden Dissertation ist aus diesem Grund das Evaluieren von bekannten und das Erarbeiten von neuen Problemeigenschaften für das DVRP. Auf der Grundlage dieser wird es erstmalig möglich eine Vielzahl von Probleminstanzen voneinander zu unterscheiden und anhand von Gemeinsamkeiten oder Unterschieden zu gruppieren. Die Charakterisierung von Probleminstanzen ist die Grundvoraussetzung für einen fairen Vergleich von Lösungsansätzen und erlaubt zusätzlich die gezielte Generierung von Testinstanzen für Algorithmen. Das Wissen über das Verhalten von Lösungsansätzen bei bestimmten Ausprägungen von Eigenschaften erlaubt zusätzlich die zielgerichtete Konstruktion von Lösungsansätzen. In der vorliegenden Dissertation wird ein Algorithmus erarbeitet, der eine sehr hohe Lösungsqualität für Probleminstanzen mit bestimmten Ausprägungen von Problemeigenschaften erzielt.

Die neu eingeführten Eigenschaften erlauben eine genaue Charakterisierung von Instanzen und bilden damit auch die Voraussetzung für die Umsetzung des in Abbildung 1.4 vorgestellten Ansatzes. Die betrachteten Problemeigenschaften basieren teilweise auf Informationen über die dynamischen Komponenten einer DVRP-Instanz. In einer realen Problemstellung ist über diese zukünftigen Aspekte vor dem Ausrollen einer Problemlösung jedoch nichts bekannt. Wird eine Auswahl von Algorithmen genau auf diesen Eigenschaften realisiert, können die dynamischen Aspekte identifiziert werden, die einen großen Einfluss auf die Lösungsqualität von Algorithmen haben. Diese Erkenntnisse können so für die zielgerichteten Erstellung von Prognosemodellen über DVRP genutzt werden. Ohne diese Modelle, die, zum Beispiel, auf der Grundlage von historischen Daten oder Expertenwissen erstellt werden können, ist eine Selektion von Algorithmen nicht möglich. Die vorliegende Arbeit fokussiert nicht die Erstellung der Prognosemodelle. Sie unterstützt bei der Identifikation notwendiger Modelle und zeigt, dass auf der Basis dieser eine erfolgreiche Selektion von Algorithmen möglich ist. Gerade für Online-Probleme ist die Selektion von Lösungsansätzen essenziell, da online getroffene Entscheidungen nicht rückgängig gemacht werden können. Die vorliegende Arbeit identifiziert die Eigenschaften, die einen bedeutenden Einfluss auf die Lösungsqualität, also auf die Beantwortung der Frage nach dem besten Algorithmus haben. Es wird gezeigt, dass auf der Grundlage dieser Eigenschaften eine manuelle und automatisierte Selektion von Algorithmen mithilfe von Methoden des maschinellen Lernens auch für Online-Optimierungsprobleme am Beispiel des DVRP erfolgreich ist. Die Arbeit legt damit unter anderem die Grundlage für die zielgerichtete Erstellung von Vorhersagemodellen für DVRP, die die Auswahl von geeigneten Algorithmen für reale Problemstellungen erlauben. In Anwendungsfällen, für die keine Prognosemodelle erstellt werden können, müssen Algorithmen eingesetzt werden, die für möglichst viele

unterschiedliche Problemstellungen gute Ergebnisse erzielen. Für die Identifikation solcher Algorithmen müssen Testinstanzen zur Verfügung stehen, die verschiedenste Eigenschaften aufweisen. Auch in solch einem Fall sind die in dieser Arbeit eingeführten Problemeigenschaften unverzichtbar. Bei der automatisierten Selektion von Algorithmen verbergen sich Erkenntnisse über den Zusammenhang zwischen Problemschwere und Beschaffenheit von Instanzen im Optimierer. Bei der Erarbeitung der Grundlagen für die manuelle Selektion werden diese Erkenntnisse aufgearbeitet und so auch unabhängig des implementierten Optimierers, über diese Arbeit hinaus, nutzbar.

Ein Optimierer, der die Frage nach dem besten Lösungsansatz für konkrete Problemstellungen beantwortet, muss ein Verständnis über die Beziehungen zwischen Aufbau und Struktur einer Probleminstanz und der Lösungsqualität von Algorithmen entwickeln. Die Anwendung dieses Wissens ist die Grundlage für die erfolgreiche Auswahl des geeignetsten Ansatzes. Die Datenbasis, aus der solches Wissen extrahiert werden kann, wird in der vorliegenden Arbeit mithilfe eines umfangreichen Data-Farming-Experiments erzeugt. Ein Probleminstanzgenerator realisiert einen Genetischer Algorithmus (GA) und generiert zielgerichtet DVRP-Instanzen mit besonders ausgeprägten Problemeigenschaften. Insgesamt werden so über eine Million unterschiedliche dynamische Problemstellungen erzeugt, die mithilfe von mehreren hundert Millionen Simulationen evaluiert werden. Aufgrund fehlender frei verfügbarer Simulatoren, wird für die Evaluation der Instanzen ein Simulationsmetamodell für DVRP mit zugehörigem Simulator vorgestellt. Mithilfe des nach einer aktuellen Taxonomie klassifizierten Metamodells können eine Vielzahl von verschiedenen Variationen des DVRP als Modell realisiert werden. Der implementierte Simulator ist unter der Apache-Lizenz Version 2.0 frei verfügbar und stellt umfangreiche Schnittstellen für Planungs- und Anpassungsalgorithmen zur Verfügung. Der Planungsalgorithmus verarbeitet a priori Probleminformation. Der Anpassungsalgorithmus reagiert auf die sich über die Zeit ändernden Informationen. Abbildung 1.5 zeigt die Integration der Simulation in den Lösungsansatz aus Abbildung 1.4 und spezialisiert den erarbeiteten Ansatz für das Online-Optimierungsproblem DVRP.

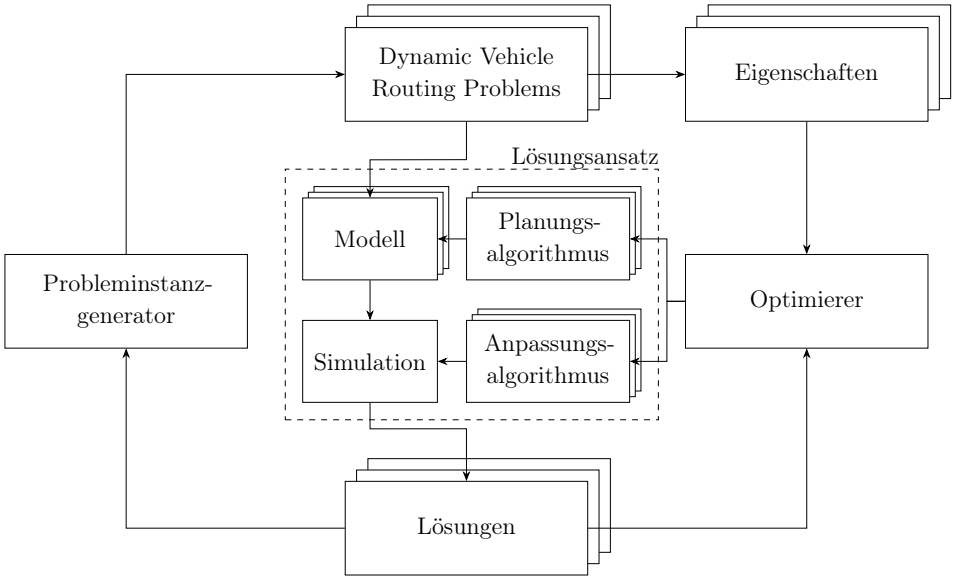

Abbildung 1.5: Spezialisierung des in Abbildung 1.4 vorgestellten Ansatz für das DVRP

Die nachfolgende Zielhierarchie fasst die wissenschaftlichen Zielstellungen, die in der vorliegenden Arbeit umgesetzt werden, zusammen. Die Ziele motivieren sich aus der Forschungsfrage nach dem geeignetsten Algorithmus für eine konkrete Probleminstanz. Die Reihenfolge der Ziele orientiert sich am Aufbau der Arbeit.

Selektion von Algorithmen für Online-Optimierungsprobleme am Beispiel des DVRP

1. Evaluation vorhandener Modelle und Simulatoren, Implementierung eines Simulationsmetamodells und eines Simulators

2. Bewertung bekannter Eigenschaften und Einführung neuer Problemeigenschaften

3. Konstruktion und Umsetzung eines Algorithmus für Instanzen mit bestimmten Ausprägungen von Eigenschaften

4. Konzeption und Implementierung eines Ansatzes zum zielgerichteten Konstruieren von Probleminstanzen

5. Untersuchung der Beziehungen und Zusammenhänge zwischen Problemeigenschaften und Lösungsqualität von Algorithmen und damit Erarbeitung der Grundlagen für die manuelle Selektion von Algorithmen

6. Umsetzung und Evaluation eines Optimierers, der auf der Grundlage von gelernten Zusammenhängen zwischen Lösungsqualität und Problemeigenschaften automatisiert Entscheidungen für Algorithmen mit hoher Lösungsqualität bei gegebenen Problemeigenschaften trifft

Grundsätzlich kann der in Abbildung 1.4 vorgestellte Ansatz auf verschiedene Problemstellungen übertragen werden (Smith-Miles, 2009). Die aufgelisteten Zielstellungen beschreiben ein Vorgehensmodell, das die Umsetzung des Ansatzes in beliebigen Problemfeldern unterstützen und anleiten kann.

1.3 Forschungsvorgehen und Aufbau der Arbeit

Die mit dieser Arbeit adressierte Forschungsfrage ist zum Teil ein klassischer Anwendungsfall für die empirische Forschung. Im Gegensatz zum Rationalismus geht die empirische Forschung davon aus, dass Erkenntnisse auf der Grundlage von Wahrnehmungen gewonnen werden und Wissen aus Erfahrungen entsteht (Beller, 2016). Die Auswahl eines Algorithmus basiert auf dem Wissen über die Beziehungen zwischen Problemschwere und Problemeigenschaften. Dieses Wissen wird aus einer Datenbasis extrahiert, die durch ein umfangreiches Data-Farming-Experiment entsteht. Es werden in dieser Arbeit also quantitative Methoden verwendet, um Erfahrungen zu generieren. Aus der erhobenen Datenbasis werden Problemeigenschaften identifiziert, die einen wesentlichen Einfluss auf die Schwere von Algorithmen haben. Zusätzlich werden Erkenntnisse abgeleitet, auf deren Grundlage eine manuelle Selektion von Algorithmen ermöglicht wird. Der im Zuge dieser Arbeit entwickelte Optimierer, der automatisiert Wissen aus der Datenbasis erarbeitet und anwendet, ist aber eher als ein neues und innovatives Artefakt zu verstehen und wird aus diesem Grund mithilfe des von Hevner et al. (2004) vorgestellten konstruktionsorientierten Forschungsansatzes (Design-Science) entwickelt. Für das Ableiten von Wissen aus Erfahrungen müssen im Zuge dieser Arbeit Problemeigenschaften eingeführt werden. Zusätzlich werden für die Realisierung des Data-Farming-Experiments ein DVRP Simulator und ein DVRP-Algorithmus eingeführt. Hierbei handelt es sich ebenfalls um neue und innovative Artefakte. All diese Artefakte können unabhängig voneinander existieren.

Im Folgenden werden die fünf zu realisierenden Schritte einer empirischen Forschung nach Beller (2016) vorgestellt. Kurz wird diskutiert, in welchen Teilen der Arbeit die Umsetzung der Schritte für die Realisierung der manuellen Auswahl von Algorithmen beschrieben ist.

1. Schritt: Formulieren und Präzisieren der Forschungsfrage

    - Die Forschungsfrage wird in Kapitel 1 anhand von motivierenden Beispielen formuliert. Im weiteren Verlauf der Einleitung wird die Fragestellung in Abschnitt 1.1 präzisiert. Abschnitt 1.2 präsentiert die aus den Fragestellungen abgeleitete Zielstellungen für diese Arbeit. Zu beachten ist, dass die empirische Forschung nur für die Realisierung eines Teils der formulierten Ziele angewendet wird. Im Vordergrund steht hier die manuelle Selektion von Algorithmen, die auf der Grundlage von Wissen über Beziehungen und Zusammenhängen zwischen Problemeigenschaften und Lösungsqualität von Algorithmen durchgeführt werden kann.

2. und 3. Schritt: Planung und Vorbereitung der Datenerhebung, Datenerhebung

   - Der Planung und der Vorbereitung der Datenerhebung widmet sich Kapitel 3 und Kapitel 5. Kapitel 3 beschreibt das entwickelte Artefakt DVRP Simulator, der die Grundlage für die Evaluation von Probleminstanzen darstellt. In Abschnitt 5.1 wird der Plobleminstanzgenerator vorgestellt und das durchgeführte Data-Farming-Experiment erläutert.

4. und 5. Schritt: Datenauswertung und Analyse, Visualisierung der Ergebnisse

   - Mit dem 4. und 5. Schritt setzt sich Abschnitt 5.2 auseinander. Eine erste Auswertung und Analyse der Daten findet bereits in Unterabschnitt 5.1.3 statt. Hier wird vor allem die Weiterverwendbarkeit der erzeugten Datenbasis untersucht. Unterabschnitt 5.2.1 identifiziert dann die Eigenschaften, die die größte Auswirkung auf die Lösungsqualität von Algorithmen haben. Anhand dieser Eigenschaften werden in Unterabschnitt 5.2.2 schwere und leichte Problemstellung gegenübergestellt. Die Gegenüberstellung erlaubt das Ableiten von Erkenntnissen, auf deren Grundlage eine manuelle Selektion von Algorithmen ermöglicht wird.

Für die Erstellung der innovativen Artefakte DVRP-Simulator, Problemeigenschaften, Algorithmus und Optimierer wird ein konstruktionsorientierter Forschungsansatz gewählt. Die sechs Aktivitäten der gestaltungsorientierten Forschung nach Peffers et al. (2007) werden mit Bezug zur Arbeit im Folgenden vorgestellt. Die Reihenfolge der Bearbeitung der Aktivitäten muss nicht eingehalten werden.

1. Motivation und Problemidentifikation

   - Die bestimmende Forschungsfrage für die vorliegende Arbeit wird in Kapitel 1 identifiziert und motiviert. Im weiteren Verlauf der Einleitung wird die Fragestellung in Abschnitt 1.1 präzisiert. Abschnitt 1.2 präsentiert die aus den Fragestellungen abgeleitete Zielstellungen für diese Arbeit. Mithilfe der gestaltungsorientierten Forschung werden die Artefakte DVRP-Simulator, Problemeigenschaften, Algorithmus und Optimierer adressiert.

2. Anforderungen der Lösungen definieren

   - Für alle umgesetzten Artefakte werden verschieden Anforderungen ausgehend von einer intensiven Literaturrecherche in Kapitel 2 definiert. Abschnitt 1.2 erläutert mithilfe der Ableitung von konkreten Zielen aus der wissenschaftlichen Fragestellung die ersten Anforderungen an einen umzusetzenden Optimierer. Kapitel 3 widmet sich dem entwickelten DVRP-Simulator. Hier werden in der Einleitung des Kapitels konkrete Anforderungen beschrieben. Kapitel 4 führt die Problemeigenschaften und den Algorithmus ein. Die Anforderungen an die Artefakte werden in den Einleitungen in Abschnitt 4.1 und Unterabschnitt 4.2.7 formuliert.

3. Design und Entwicklung

   - Das Design und die Entwicklung der Artefakte sind in den jeweiligen Kapiteln beschrieben. Kapitel 3 setzt sich mit dem DVRP-Simulator auseinander. In Abschnitt 4.1 werden die Problemeigenschaften eingeführt. Zusätzlich widmet sich das Kapitel 4 in Unterabschnitt 4.2.7 der Erarbeitung des DVRP-Algorithmus. Verschiedene Umsetzungen eines Optimierers werden in Abschnitt 6.1 diskutiert.

4. Demonstration und 5. Evaluation

   - Mit der Durchführung des Data-Farming-Experiments wird gezeigt, dass die entwickelten Artefakte DVRP-Simulator, Algorithmus und Eigenschaften für verschiedene Probleminstanzen anwendbar sind. In Abschnitt 5.1 wird das durchgeführte Experiment im Detail erläutert. Die Auswertung und die Analyse der erzeugten Datenbasis in Abschnitt 5.2 evaluiert die erzeugten Artefakte und bringt die Ergebnisse in Bezug zu den formulierten Anforderungen. Die Analyse in Abschnitt 5.1 zeigt, dass der eingeführte DVRP-Algorithmus einen Teil der betrachteten Probleminstanzen erfolgreicher im Vergleich zu weiteren betrachteten Algorithmen löst. Zusätzlich wird in Unterabschnitt 5.2.2 gezeigt, dass eine manuelle Selektion von Algorithmen auf der Grundlage der eingeführten Eigenschaften möglich ist. Die erfolgreiche Durchführung der automatisierten Selektion von Algorithmen mithilfe von Methoden des maschinellen Lernens, erläutert in Kapitel 6, demonstriert auch die erfolgreiche Umsetzung des Optimierers. Die hohe Qualität des Optimierers wird mit bekannten, aber auch unbekannten Probleminstanzen nachgewiesen.

6. Kommunikation

   - Im Kontext dieser Dissertation wurden Teilergebnisse durch den Autor dieser Arbeit bereits kommuniziert. Eine Vorgängerversion des erarbeiteten DVRP-Simulators wurde, zum Beispiel, in Mayer et al. (2016) vorgestellt. Mayer et al. (2017) und Mayer et al. (2018*b*) widmen sich den neu eingeführten Problemeigenschaften. In Mayer et al. (2018*a*) kann der Autor dieser Arbeit eine erfolgreiche Implementierung eines Optimierers für die automatisierte Auswahl von Algorithmen für das DVRP vorstellen. Auf die Veröffentlichungen im Kontext dieser Dissertation vom Autor dieser Arbeit wird in den inhaltlich passenden Abschnitten Bezug genommen.

Abbildung 1.6 veranschaulicht den Aufbau der vorliegenden Arbeit grafisch mithilfe des umgesetzten Ansatzes visualisiert in Abbildung 1.5. Jedes folgende Kapitel nutzt die Visualisierung, um den beschriebenen Inhalt in den Kontext dieser Arbeit einzuordnen. Im folgenden Kapitel 2 werden die wissenschaftlichen Grundlagen gelegt, die für die

Umsetzung der Zielstellung benötigt werden. Der Fokus liegt hier auf der Erarbeitung des DVRP und des ASP. Sollten dem Leser diese Grundlagen bereits bekannt sein, kann das Studium von Abschnitt 2.1 und 2.2 übersprungen werden. In Abschnitt 2.3 wird die in dieser Arbeit betrachtete Variation des DVRP abgegrenzt. Zusätzlich werden verwendete Vokabeln erläutert und häufig gebrauchte Synonyme eingeführt. Für das Verständnis der weiteren Kapitel ist Abschnitt 2.3 elementar und sollte aus diesem Grund nicht übersprungen werden.

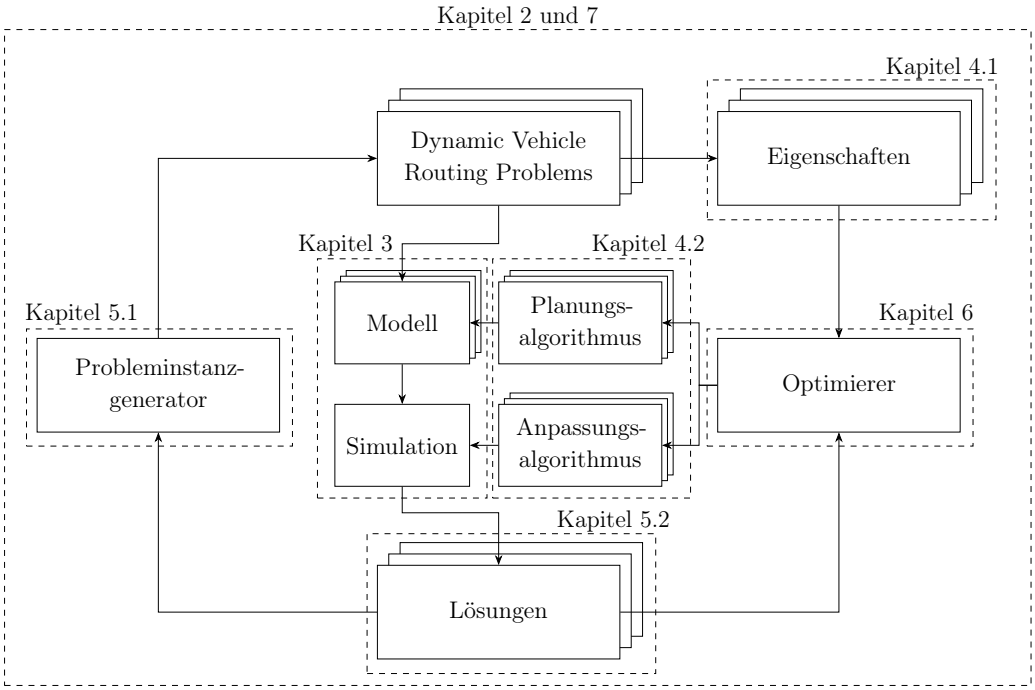

Abbildung 1.6: Visualisierung der Grobgliederung der vorliegenden Arbeit mithilfe des umgesetzten Ansatzes

Kapitel 3 setzt sich mit dem implementierten DVRP-Simulator auseinander. Abschnitt 3.1 erläutert den Simulatorkern und Abschnitt 3.2 konzentriert sich auf das implementierte Simulationsmetamodell und die Klassifikation dessen. Das Kapitel kann losgelöst von der restlichen Arbeit studiert werden. Sollte der Leser ausschließlich an dem Simulator interessiert sein und kurz überprüfen wollen, ob dieser den Anforderungen der eigenen Problemstellung genügt, sei auf Unterabschnitt 3.2.4 und die Übersichtstabelle 3.3 verwiesen. Hier werden die vom Simulator unterstützten Varianten des DVRP zusammengefasst. In Kapitel 4 werden Problemeigenschaften für das DVRP eingeführt und unabhängig von Algorithmenselektion evaluiert (Abschnitt 4.1). Zusätzlich werden die betrachteten Algorithmen vorgestellt (Abschnitt 4.2). Der im Kontext dieser Arbeit eingeführte Algorithmus ist Teil der Beschreibung der betrachteten Algorithmen und wird in Unterabschnitt 4.2.7 erläutert. Für die Umsetzung der manuellen und automatisierten Selektion von Algorithmen sind die Problemeigenschaften und Lösungsansätze elementar. Für das Verständnis des umgesetzten Ansatzes ist das Studium von Kapitel 4 hingegen nicht notwendig. Kapitel 5 legt den Fokus auf die Exploration und die

Analyse von Problemen, Lösungen und Eigenschaften. Hier wird in Abschnitt 5.1 der Probleminstanzgenerator vorgestellt und das umfangreiche Data-Farming-Experiment diskutiert. Eine erste Evaluation der erzeugten Datenbasis in Unterabschnitt 5.1.3 weist die Eignung der Daten zur Weiterverwendung nach. In Abschnitt 5.2 wird jetzt die Wichtigkeit der Problemeigenschaften untersucht und die Beziehungen zwischen den einflussreichsten Eigenschaften und der Problemschwere hergeleitet. Die in Abschnitt 5.2 erarbeiteten Erkenntnisse dienen als Grundlage für die manuelle Selektion von Algorithmen. Mit der Realisierung von Selektionsmodellen beschäftigt sich Kapitel 6. Hier erfolgt auch der Nachweis der erfolgreichen Auswahl von Algorithmen durch die implementierten Modelle. In weiterführenden Experimenten wird in Unterabschnitt 6.1.3 gezeigt, dass die implementierten Modelle auf der Grundlage der erzeugten Datenbasis auch in der Lage sind, Algorithmen erfolgreich für vollständig unbekannte Probleminstanzen zu selektieren. Die Diskussion in Abschnitt 6.2 schließt das Kapitel 6. Da die Diskussion unter anderem die Leistungsfähigkeit der Selektionsmodelle erläutert, werden hier auch Bezüge zu den durchgeführten Experimenten, erläutert in Kapitel 5, hergestellt. Kapitel 5 und 6 sollten aus diesem Grund in der gegebenen Reihenfolge und nicht unabhängig voneinander studiert werden. In Kapitel 7 werden abschließend die Ergebnisse der vorliegenden Dissertation zusammengefasst und das Erreichte anhand der Zielstellungen aus Abschnitt 1.2 erläutert und eingeschätzt. Kapitel 7 schließt mit einem Ausblick und der Diskussion möglicher Weiterentwicklungen.

In der gesamten Arbeit werden für verschiedene Erläuterungen DVRP-Instanzen visualisiert. In allen Visualisierungen werden statische Kundenanfragen in Form von Kreisen und dynamische Anfragen in Form von Dreiecken veranschaulicht. Liegt der Fokus der Visualisierung auf der Verteilung der Anfragen, werden für einen hohen Kontrast die Kreise grau und die Dreiecke schwarz eingefärbt. Die linke Probleminstanz in Abbildung 1.7 zeigt eine Beispielprobleminstanz, die die Verteilung der Anfragen fokussiert. Die rechte Instanz, dargestellt in Abbildung 1.7, zeigt neben den Kundenanfragen zusätzlich geplante (gestrichelte Linien zwischen Anfragen) und bereits durchgeführte Teilrouten (durchgezogene Linien zwischen Anfragen). Die rechte Probleminstanz in Abbildung 1.7 fokussiert nicht die Verteilung der Anfragen. Statische Kunden sind hier mit schwarzen Kreisen und dynamische mit unausgefüllten Dreiecken visualisiert. Die in Abbildung 1.7 dargestellten Probleminstanzen stehen exemplarisch für Visualisierungen von DVRP-Instanzen in dieser Arbeit. Die Bedeutung der gewählten Symbole wird in zukünftigen Visualisierungen nicht mehr näher erläutert.

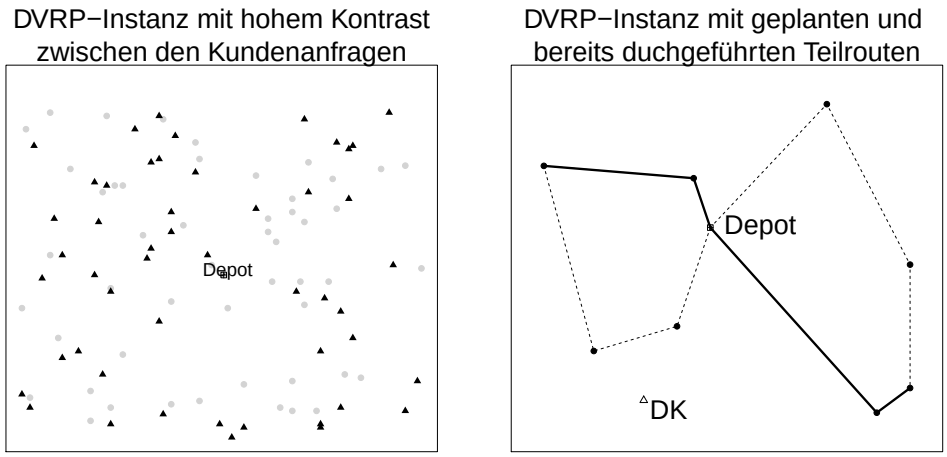

Abbildung 1.7: Exemplarische DVRP-Instanzen mit dynamischen und statischen Kunden, geplanten und bereits durchgeführten Teilrouten

Kapitel 2

Methoden und Grundlagen

In Kapitel 2 werden die wissenschaftlichen Grundlagen für die gesamte vorliegende Arbeit erläutert. Abbildung 2.1 zeigt aus diesem Grund die Visualisierung der umgesetzten Lösung mit dem Fokus auf allen Komponenten. In Abschnitt 2.1 wird das *Dynamic Vehicle Routing Problem* (DVRP) als Spezialfall des *Vehicle Routing Problem* (VRP) diskutiert. Es werden bekannte Metriken zur Berechnung von Problemeigenschaften vorgestellt und es kann anschaulich demonstriert werden, dass eine Vielzahl von Instanzen mit diesen existierenden Metriken nicht zu unterscheiden sind. In Abschnitt 2.1 werden Lösungsansätze für das DVRP erläutert und es wird gezeigt, dass diese über existierende Problemgruppen hinweg erfolgreich angewendet werden. Zusätzlich werden in Abschnitt 2.1 Benchmarkproblemstellungen und Modelle für die Abbildung von VRP diskutiert.

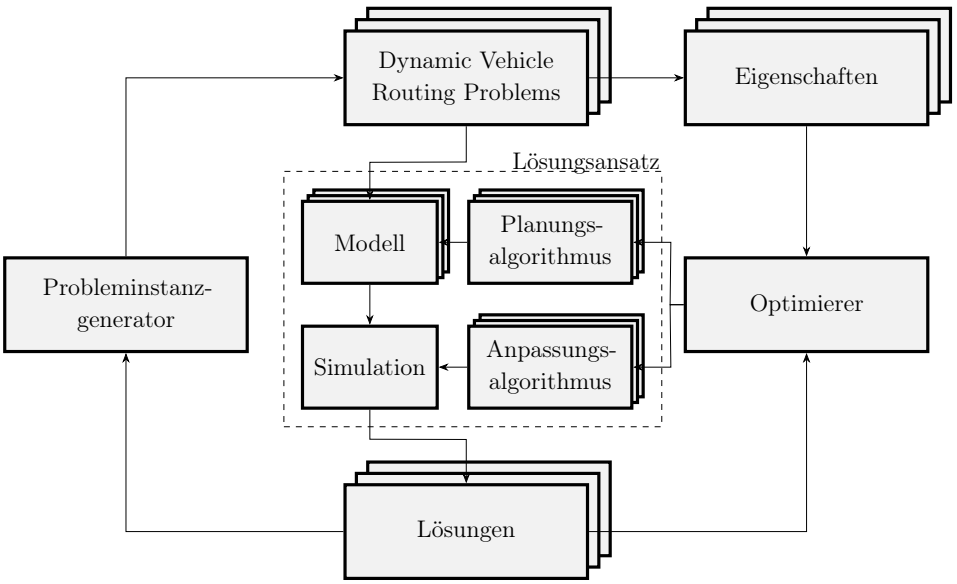

Abbildung 2.1: Lösungsansatz mit Fokus auf allen Komponenten

Der Abschnitt 2.2 behandelt das *Algorithm Selection Problem* (ASP). Nach einer kurzen Einführung werden die Komponenten des Frameworks für das ASP vorgestellt.

Mit der Erläuterung der Komponenten werden Metriken zur Berechnung von Eigenschaften des *Travelling Salesman Problem* (TSP) eingeführt, die für die Selektion von Algorithmen verwendet werden und die Ausgangsbasis für die im Zuge dieser Arbeit entwickelten Metriken zur Berechnung von Problemeigenschaften für das DVRP sind. Zusätzlich werden Anwendungsbeispiele auch in der Domäne Fahrzeugwegeplanung erläutert und die theoretischen Grundlagen für die Erstellung von Selektionsmodellen diskutiert.

2.1 Das Dynamische Fahrzeugwegeplanungsproblem

Das *Dynamic Vehicle Routing Problem* (DVRP) (dt. dynamisches Fahrzeugwegeplanungsproblem) ist ein Spezialfall des *Vehicle Routing Problem* (VRP). Es unterscheidet sich hinsichtlich der Evolution der Probleminformation. Bei einem statischen VRP ändert sich im Gegensatz zu einem DVRP die Probleminformation nicht. Im folgenden Unterkapitel wird das VRP eingeführt. Mithilfe einer aktuellen Klassifikation werden zusätzlich eine Vielzahl von Problemvariationen vorgestellt. Unterabschnitt 2.1.2 beschäftigt sich im Detail mit den Besonderheiten der dynamischen Variante des VRP. Im Anschluss werden Methoden zum Erfassen der Dynamik mithilfe von Metriken diskutiert. Es wird erarbeitet, dass die vorhandenen Möglichkeiten zur Charakterisierung von Probleminstanzen eines DVRP nicht ausreichend sind.

Das Kapitel setzt sich ebenfalls mit typischen Anwendungsbeispielen auseinander und ordnet diese in bekannte Problemgruppen ein. Neben Anwendungen, werden auch Lösungsansätze diskutiert, die häufig Anwendung finden. Durch die Zuordnung von Anwendungen in Kombination mit einem Lösungsansatz auf Problemgruppen kann gezeigt werden, dass Lösungsansätze gruppenübergreifend zum Einsatz kommen, also eine Auswahl von einem Lösungsansatz auf Basis von bekannten Problemgruppen nicht erfolgen kann. Durch die Vorstellung von Methoden zum Bewerten der Qualität von Problemlösungen wird in Unterabschnitt 2.1.6 gezeigt, dass für den Vergleich von Algorithmen die betrachteten Methoden erweitert werden müssen. Eine entsprechende Erweiterung wird ebenfalls in Unterabschnitt 2.1.6 diskutiert. Das Kapitel schließt mit einem kurzen Überblick über vorhandene Benchmarkproblemstellungen.

2.1.1 Fahrzeugwegeplanung

Das *Vehicle Routing Problem* (VRP) (dt. Fahrzeugwegeplanungsproblem) wurde erstmals von Dantzig und Ramser (1959) am Beispiel von Benzinlieferungen eingeführt. Es ist ein sehr generisches Problem, das sich im Wesentlichen mit der Planung von Routen für eine Flotte von Fahrzeugen zu Kunden beschäftigt. Das VRP ist eine Verallgemeinerung des TSP, dass sich mit der Planung von genau einer Rundreise für ein Fahrzeug beschäftigt. Im Gegensatz zum *Chinese Postman Problem* (CPP), eingeführt in Mei-Ko (1962), das sich mit dem Besuch aller Wege zwischen Knoten bzw. Kunden beschäftigt (Grötschel und Yuan, 2012), konzentriert sich das TSP auf das Besuchen

aller Kunden. Das TSP wurde unter anderem untersucht und formuliert von Flood (1956). Das VRP ist ein NP-hartes kombinatorisches Optimierungsproblem (Lenstra und Kan, 1981) und wird aus diesem Grund üblicherweise heuristisch gelöst (Grötschel und Padberg, 1977). Das VRP lässt sich mithilfe des Graphen $G(V,E,C)$ formulieren. Dabei ist $V = \{v_0, v_1, ..., v_n\}$ eine Menge von Knoten. Üblicherweise repräsentiert der Knoten v_0 das Depot, der initiale Standort der Fahrzeugflotte. Die Knoten $V' = V \setminus \{v_0\}$ definieren die Standorte der Kunden (oder Anfragen), zu denen Routen geplant werden müssen. Eine Beispielprobleminstanz ist in Abbildung 2.2 im linken Bild dargestellt. Jeder Knoten v_i in V' kann eine Menge an zu liefernden Gütern q_i definieren. Die Menge $E = \{(v_i,v_j)|v_i,v_j \in V; i \neq j\}$ definiert alle benachbarten Knoten. Die beim Bewegen zwischen den benachbarten Knoten v_i und v_j entstehenden Kosten $c_{i,j}$, werden mithilfe einer Kostenmatrix C definiert. Bei den Kosten kann es sich, zum Beispiel, um Distanzen, Fahrzeiten oder Reisekosten handeln. Wenn $c_{i,j} = c_{j,i}$ für alle $(v_i, v_j) \in V$ handelt es sich um ein symmetrisches VRP. Sind $c_{i,j} \neq c_{j,i}$ für mindestens eine Kombination von (i,j) in $(v_i, v_j) \in V$, ist das VRP asymmetrisch.

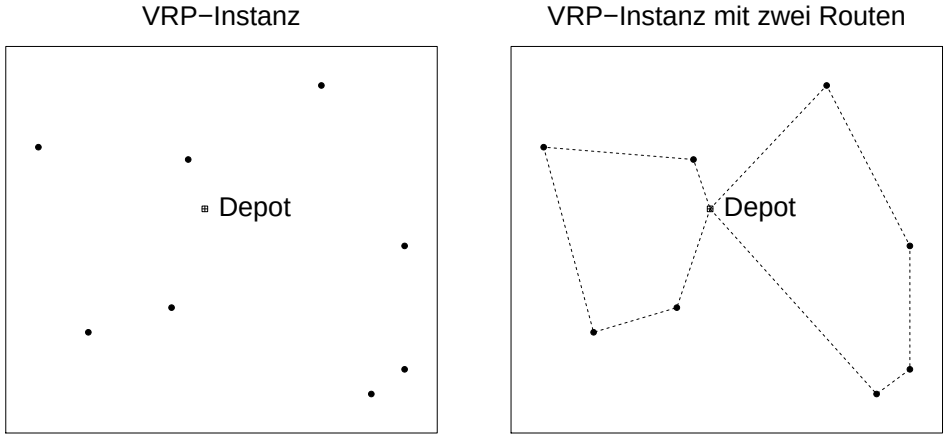

Abbildung 2.2: Visualisierung einer VRP Beispielprobleminstanz ohne (linkes Bild) und mit geplanten Teilrouten (rechtes Bild)

Alle m identischen Fahrzeuge der Flotte werden mit einer Kapazität k charakterisiert. Für das TSP, als Spezialfall des VRP bei dem nur ein Fahrzeug für die vollständige Routenplanung benötigt wird, gilt $k \geq \sum_{v_i \in V} q_i$. Eine Lösung für das VRP definiert eine Menge von Routen $R_1,..., R_m$ für die Fahrzeuge der Flotte. Eine Route R_i für das Fahrzeug i startet und endet üblicherweise am Depot v_0 und definiert eine Menge von Knoten, die durch das zugeordnete Fahrzeug bedient werden. Die Kosten einer Route $(R_i = \{v_0, v_1,..., v_{m+1}\})$, bei dem $v_i \in V$ und $v_0 = v_{m+1} = 0$ (0 entspricht dem Depot), sind definiert durch $C(R_i) = \sum_{i=0}^{m} c_{i,i+1}$. Die Kosten der Problemlösung ergeben sich aus der Summe der Kosten der Routen. Eine Beispielprobleminstanz mit zwei möglichen Routen ist in Abbildung 2.2 im rechten Bild dargestellt.

Neben dieser klassischen Problemformulierung existieren viele weitere Problemvariationen. Diese beschreiben oft Ziele, Nebenbedingungen oder Unsicherheiten, die in

realen Problemstellungen berücksichtigt werden müssen. Häufig werden diese Probleme als *Rich VRP* (RVRP) bezeichnet. Eine eindeutige Definition von RVRP gibt es jedoch nicht (Ulmer, 2017). Lahyani et al. (2015) ordnet ein VRP mithilfe der Menge an bestimmten Nebenbedingungen der Menge der RVRP zu. Dafür führt Lahyani et al. (2015) eine umfangreiche Taxonomie für das VRP ein. Die Taxonomie von Lahyani et al. (2015) unterscheidet dabei grundlegend zwischen Szenario-Eigenschaften und Problem-Physikalischen-Eigenschaften einer Routingproblemstellung. Alle Kategorien der Taxonomie mit kurzen Erläuterungen sind in der folgenden Aufzählung zusammengefasst und werden im Anschluss detailliert erläutert.

1. **Szenario-Eigenschaften**
    - 1.1. **Eingabedaten** - Unterscheidung hinsichtlich Evolution und Qualität der Eingabedaten
    - 1.2. **Entscheidungsmanagement** - Übergeordnete Entscheidungen die Grundlage der Routenplanung sein können
    - 1.3. **Anzahl der Depots** - Anzahl und Art der Depots
    - 1.4. **Operationstyp** - Kategorisiert die Art der Kundenanfragen
    - 1.5. **Ladungsteilung Restriktionen** - Können einzelne Anfragen durch unterschiedliche Routen bzw. Fahrzeuge bedient werden
    - 1.6. **Planungsperiode** - Beschreibung des betrachteten Zeithorizonts
    - 1.7. **Mehrfachnutzung der Fahrzeuge** - Ist eine Mehrfachnutzung der Fahrzeuge vorgesehen
2. **Problem-Physikalische-Eigenschaften**
    - 2.1. **Fahrzeuge** - Kategorisierung der Flotte (Anzahl, Typ und Struktur), die Flotte schließt hier auch mögliche Fahrer und Restriktionen hinsichtlich dieser ein
    - 2.2. **Zeitbezogene Restriktionen** - Werden Zeitfenster, Servicezeiten oder Wartezeiten betrachtet und welche Problemkomponenten werden mit solchen Restriktionen versehen
    - 2.3. **Zeitfensterstruktur** - Beschaffenheit der Zeitfenster
    - 2.4. **Inkompatibilitäten** - Restriktionen, die den Transport von Waren und Gütern oder das Erfüllen von Serviceleistungen einschränken.
    - 2.5. **Spezifische Restriktionen** - Weitere Restriktionen
    - 2.6. **Zielfunktion** - Kategorisierung der Zielfunktion

Eingabedaten

Besonders in realen Anwendungsfällen des VRP ist die Evolution und Qualität der Probleminformationen (siehe Aufzählung 1.1.) eine entscheidende Problemcharakteristik (Psaraftis, 1980). Teilweise werden Routenplanern Probleminformationen erst während der Durchführung bekannt, zum Beispiel, mit Anfragen von neuen Kunden. Auch schwankt die Qualität der Information in einem realen Anwendungsfall stark. Vor allem Transportzeiten unterliegen üblicherweise hohen Unsicherheiten. So ergeben sich unter anderem nach Pillac et al. (2013) die vier in Tabelle 2.1 dargestellten Probleminformationskategorien.

Tabelle 2.1: VRP Kategorien hinsichtlich Evolution und Qualität der Probleminformation nach Pillac et al. (2013)

		Informationsqualität	
		Deterministische Information	Stochastische Information
Informations-evolution	Informationen bekannt	Statisch und deterministisch	Statisch und stochastisch
	Information verändert sich über die Zeit	Dynamisch und deterministisch	Dynamisch und stochastisch

In **statischen und deterministischen** Problemstellungen sind alle Probleminformationen a priori bekannt und unterliegen keinen Unsicherheiten. Einen aktuellen Überblick über Forschungen in dieser Problemkategorie liefert, zum Beispiel, Laporte (2009). In **statischen und stochastischen** Problemstellungen sind zwar alle Probleminformationen a priori bekannt, unterliegen aber stochastischen Schwankungen. Allgemein können alle Eingabedaten diesen Schwankungen unterliegen. Oft werden Unsicherheiten bei Kunden (Kunden treten nur mit bestimmter Wahrscheinlichkeit auf), Zeiten (schwankende Fahr- und Servicezeiten) und Anfragen (schwankende Menge der zu liefernden Güter) beschrieben. Auch Verfügbarkeiten von Fahrzeugen und Fahrern unterliegen in verschiedenen Problembeschreibungen Unsicherheiten (Mu et al., 2011). Probleme dieser Kategorie werden in der Literatur häufig als *Stochastic VRP* (SVRP) eingeführt. Einen aktuellen und umfangreichen Überblick über diese Problemkategorie und eine Taxonomie beschreibt Berhan et al. (2014). Bei **dynamisch und deterministischen** Problemstellungen unterliegen die Eingabedaten keinen Unsicherheiten, Probleminformationen ändern sich aber über die Zeit. Üblicherweise ist hier das Erscheinen neuer Kunden (oder Anfragen) gemeint. Probleme dieser Kategorie werden in der Literatur häufig als DVRP eingeführt und stehen im Fokus dieser Arbeit und werden im folgenden Unterkapitel genauer beleuchtet. Umfangreiche Reviews über diese Problemkategorie sind, zum Beispiel, in Larsen et al. (2008) und Psaraftis et al. (2016) zu finden. In **dynamischen und stochastischen** Problemstellungen unterliegen die Eingabedaten Unsicherheiten und Probleminformationen ändern sich über die Zeit. In

der Literatur werden Probleme dieser Kategorie selten auch als *Stochastic Dynamic Resource Allocation Problem* (SDRAP) eingeführt (Godfrey und Powell, 2002). Die Besonderheit ist dabei, dass in einem SDRAP Transportressourcen selbst Empfänger von Services an den Knoten (bei den Kunden) sein können. Aktuelle Entwicklungen in dieser Problemkategorie sind ausführlich beschrieben in Ritzinger et al. (2016).

Entscheidungsmanagement

Beim klassischen Vehicle Routing werden Entscheidungen auf der Grundlage von konkreten Kundenanfragen getroffen. Der Zeitpunkt, wann diese Anfragen ausgelöst werden obliegt üblicherweise dem Kunden. Bei großen und komplexen Lieferketten wird es zunehmend üblich, diese Entscheidungen auf den Zulieferer auszulagern (Arndt, 2004). Das Auslagern der Entscheidung führt dabei oft zu signifikantem Performancegewinnen in der Lieferkette (Moin und Salhi, 2007). Lagerbestand, der durch den Zulieferer kontrolliert wird, bezeichnet man in der Literatur häufig als *Vendor Managed Inventory* (VMI) oder auch als *Supplier Managed Inventory* (SMI). Das daraus resultierende Routingproblem, bei dem Entscheidungen auf der Grundlage des Verbrauchs der Kunden getroffen werden, wird in der Literatur als *Inventory Routing Problem* (IRP) bezeichnet. Eine frühe Variante des Problems beschreibt Bell et al. (1983) am Beispiel eines Produzenten von Industriegasen. Eine umfangreiche Übersicht über das IRP ist in Campbell et al. (1998) zu finden. Oft ist der Ursprung von Erweiterungen des Vehicle Routings im Lieferketten-Management (Supply Chain Management) zu finden. Eine Zusammenfassung solcher Problemstellungen, meist auch als Rich VRP bezeichnet, liefert Schmid et al. (2013).

Problemvariationen, die neben einer Routing- noch zusätzliche Entscheidungen treffen werden von Lahyani et al. (2015) auch mithilfe der Kategorie **Entscheidungsmanagement** unterschieden. Werden, zum Beispiel, Standortentscheidungen, wie die Lage von Depots oder Verteilungszentren, neben dem Routing getroffen, spricht man allgemein vom *Location Routing Problem* (LRP). Bruns (1998) definiert das LRP als 'Standortplanung unter Berücksichtigung von Tourenplanungsaspekten'. Ein umfangreiches Review über diese Problemvariation liefert, zum Beispiel, Nagy und Salhi (2007). Neben Standortentscheidungen werden unter anderem auch Personalentscheidungen zusammen mit dem Routing getroffen. So beschreibt die Lösung einer solchen Problemvariation neben einer Menge von Touren, auch den Fahrer, der die Touren durchführen soll. Das *Practical Vehicle Routing and Driver Scheduling Problem* (PVRDCP) wird unter anderem in Wen et al. (2011) diskutiert.

Anzahl der Depots

Das klassische VRP beschreibt genau ein Depot, bei dem alle Fahrzeuge stationiert sind. Meist startet und endet eine Tour bei diesem Depot. Reale Anwendungsfälle hingegen berücksichtigen oft mehrere Standorte für die Fahrzeugflotte. Mit der Lösung einer solchen Problemstellung widmet sich erstmalig Tillman (1969) unter der Bezeichnung

Multi Terminal Delivery Problem (MTDP). Die heute üblichere Bezeichnung für ein VRP mit mehr als einem Depot ist *Multi-Depot Vehicle Routing Problem* (MDVRP). Einen umfangreichen Überblick über Forschungen im Bereich des MDVRP finden sich, zum Beispiel, in Cordeau et al. (1997) und Crevier et al. (2007).

Operationstyp

Eine weitere Dimension für die Unterscheidung von VRP ist die Art der Kundenanfrage. Klassisch beschreibt die Anfrage entweder eine Lieferung oder eine Abholung. Beim *Pick-up and Delivery Problem* (PDP) oder auch *VRP with Pickups and Deliveries* (VRPPD) ist die Kundenanfrage eine Lieferung und eine Abholung. Einen umfangreichen Überblick über diese Problemvariation liefert Desaulniers et al. (2000). Ein Spezialfall des VRPPD ist das *Dial-A-Ride Problem* (DARP), bei dem Personen zwischen Knoten transportiert werden. Ein weiterer Spezialfall des VRPPD ist das *VRP with Backhauls* (VRPB), unter anderem beschrieben in Goetschalckx und Jacobs-Blecha (1989). Bei dieser Problemvariation müssen erst alle Lieferungen auf einer Tour durchgeführt werden, bevor die Abholungen erfolgen dürfen.

Ladungsteilung Restriktionen

In der klassischen Formulierung des VRP wird ein Kunde von genau einem Fahrzeug der Flotte bedient. Dror und Trudeau (1989) führen das *VRP with Split Deliveries* (SDVRP) ein und untersuchen, ob das Aufteilen der zu liefernden Waren an einen Kunden auf mehrere Fahrzeuge und damit auf verschiedene Touren einen Vorteil bei der Tourenplanung bringt. Eine Übersicht und Lösungsansätze für das SDVRP werden in Archetti und Speranza (2012) präsentiert.

Planungsperiode

Die Länge der Planungsperiode für ein VRP ist üblicherweise stark Domänenspezifisch. Klassischerweise wird dabei ein Kunde in der Planungsperiode nur einmal bedient. Beim *Periodic VRP* (PVRP), eingeführt am Beispiel von Abfallentsorgung von Beltrami und Bodin (1974), müssen Kunden in der Planungsperiode hingegen mehrfach bedient werden. Oft wird die Planungsperiode dabei in Teilperioden zerlegt. Dabei gilt für jede Teilperiode, dass alle Kunden nur einmalig in der Periode bedient werden müssen. Jede Teilperiode kann so als klassisches VRP formuliert werden. Einen umfassenden Überblick über das PVRP bietet Campbell und Wilson (2014).

Mehrfachnutzung der Fahrzeuge

Eine Lösung für ein VRP sieht die einmalige Nutzung eines Fahrzeuges in einer Planungsperiode vor. Es wird also genau eine Tour für ein Fahrzeug geplant. Fleischmann (1990) untersucht erstmalig die Vorteile bei der Planung von mehreren Touren für ein

Fahrzeug. Lösungen für diese Problemvariation werden unter anderem in Taillard et al. (1996) vorgestellt.

Fahrzeuge

Fahrzeuge sind neben den zu bedienenden Kunden und der Infrastruktur eine Hauptkomponente des VRP. Die klassische Modellierung sieht eine homogene Fahrzeugflotte vor, also m Fahrzeuge mit identischer Transportkapazität. Schon früh wurden aber auch Formulierungen, die eine heterogene Fahrzeugflotte betrachten üblich. Kirby (1959) beschreibt das *Heterogeneous fleet VRP* (HVRP), bei dem alle m Fahrzeuge eine unterschiedliche Transportkapazität k aufweisen. Einen aktuellen Überblick über Forschungen am VRP mit heterogener Fahrzeugflotte bietet, zum Beispiel, Baldacci et al. (2008). Ein Spezialfall des HVRP ist das *Truck and Trailer Routing Problem* (TTRP), bei dem die Fahrzeugflotte aus Zugmaschinen und Anhängern besteht. Kunden (Anfragen) können bei dieser Problemvariation entweder von einer Kombination aus Zugmaschine und Anhänger bedient werden, oder nur von der Zugmaschine.

Oftmals beschäftigen sich Fragestellungen im Kontext der Fahrzeugflotte mit der Dimensionierung und der Struktur der Flotte. Eine zusätzliche typische Erweiterung des klassischen VRP ist das Betrachten von verschiedenen Transportbereichen in einem Fahrzeug. So ist es, zum Beispiel, bei dem Transport von Nahrungsmitteln üblich, unterschiedlich temperierte Transportbereiche in einem Fahrzeug zu vereinen. Auch müssen oftmals Flüssigkeiten oder Müll in einem Fahrzeug, aber getrennt voneinander transportiert werden. Ein Spezialfall des VRP mit Fahrzeugen mit verschiedenen Transportbereichen ist das *Livestock Collection Problem* (LCP) eingeführt von Oppen und Løkketangen (2006). Bei dieser Problemformulierung müssen unterschiedliche Tiere, getrennt voneinander zum Schlachthaus transportiert werden. Das LCP beschreibt viele weitere Restriktionen, die den Zweck haben das Tierwohl sicherzustellen. Intensiv setzt sich Derigs et al. (2011) mit dem VRP mit Fahrzeugen mit verschiedenen Transportbereichen auseinander. Werden verschiedene Bereiche beschrieben umfasst die Problemformulierung typischerweise auch unterschiedliche Arten von Transportgut. Gerade bei einer Vielzahl von verschiedenen Gütern kann die Ladungspolitik einen wesentlichen Einfluss auf das Routing haben. Iori (2005) führt das *Loading VRP* (LoVRP) ein und betrachtet damit erstmals die Auswirkungen von unterschiedlichen Ladungspolitiken auf das Routing. Die am häufigsten beschriebenen Ladungspolitiken sind *Last In First Out* (LIFO) und *First In First Out* (FIFO). Einen aktuellen Überblick über Forschungen zum LoVRP liefert Iori und Martello (2010).

Unter der Kategorie **Fahrzeug** werden durch Lahyani et al. (2015) auch alle Regulierungen den Fahrzeugfahrer betreffend kategorisiert. Die klassische Formulierung des VRP nimmt keine Rücksicht auf Restriktionen, wie maximale Fahrzeiten oder vorgeschriebene Pausen. Eine umfangreiche Studie über Regulationen der Fahrzeugfahrer und deren Auswirkung auf das Routing wird präsentiert in Goel und Vidal (2013). Neue Problemformulierungen, den Fahrzeugfahrer betreffend, sind aktuell motiviert aus einem Trend im Konsummarkt, der Shareconomy. Shareconomy ist das Teilen

und gemeinsame Nutzen von Ressourcen aller Art (Kraus und Giselbrecht, 2015). Das gemeinsame Nutzen von Fahrzeugen ist nicht nur im Individualverkehr Trend, Firmen wie Amazon oder Uber versuchen den Transport von Gütern oder Personen mithilfe von einer eigenen Flotte und gemeinsam genutzten Fahrzeugen zu realisieren (Uber, 2019, Nicolai, 2018). Archetti et al. (2016) formuliert das *VRP with Occasional Drivers* (VRPOD), bei dem neben der regulären Fahrzeugflotte auch gelegentliche Fahrer für das Routing zur Verfügung stehen.

Zeitbezogene Restriktionen und Zeitfensterstruktur

Die am häufigsten formulierte zeitbezogene Restriktion für das VRP ist das Zeitfenster (engl. time windows). Ein Zeitfenster definiert für einen Kunden (Anfrage) ein Zeitintervall, in dem die Lieferung, Abholung oder der Service stattfinden soll. Teilweise existieren Problemformulierungen, zum Beispiel, in Polacek et al. (2004), bei denen auch für Depots Zeitfenster modelliert werden. Knight und Hofer (1968) ist eine frühe Arbeit, die sich mit Zeitfenstern für das VRP beschäftigt. Das Problem selbst wird in der Literatur oft als *VRP with Time Windows* (VRPTW) bezeichnet. Grundsätzlich wird zwischen weichen (engl. soft) und harten (engl. hard) Zeitfenstern unterschieden. Weiche Zeitfenster erlauben das Bedienen der Kunden (Anfragen) auch außerhalb des definierten Intervalls. Bei harte Zeitfenster hingegen darf das Bedienen nur im vorgegebenen Intervall erfolgen. Das VRPTW wird umfangreich, zum Beispiel, in Kallehauge et al. (2005) analysiert. Zusätzliche Zeitrestriktionen betreffen oftmals das Be- und Entladen, aber auch das Bedienen des Kunden (Anfrage). In verschiedenen Problemstellungen wird auch die Verfügbarkeit der Infrastruktur mithilfe von Zeitfenstern formuliert. So werden, zum Beispiel, in Munuzuri et al. (2005) bestimmte Gebiete für die Durchfahrt von einzelnen Fahrzeuggruppen mithilfe von Zeitfenstern gesperrt. Auf diese Weise können, zum Beispiel, eventuelle, lokal begrenzten Fahrverboten für Dieselfahrzeuge modelliert werden. Fahrverbote von Dieselfahrzeugen in Stadtkernen sind im Gespräch aufgrund der Grenzwertüberschreitungen von Stickoxiden und den Abgasmanipulationen durch die Automobilindustrie, auch bekannt als Dieselgate (Tilp und Schiefer, 2017). Eine umfangreiche Übersicht über die Modellierung und Lösung vom VRPTW ist gegeben in Gendreau und Tarantilis (2010).

Inkompatibilitäten

Die klassische Formulierung des VRP berücksichtigt keine Inkompatibilitäten zwischen den Problemkomponenten. In realen Anwendungsfällen finden sich aber unzählige solcher Unvereinbarkeiten. Zum Beispiel, können bestimmte Güter nur in bestimmten Fahrzeugen oder Transportbereichen befördert werden, Fahrzeugfahrer können nur ausgewählte Fahrzeuge führen, oder Straßen sind für verschiedene Fahrzeuge nicht geeignet. Allgemein lässt sich sagen, dass es zu Inkompatibilitäten zwischen allen Komponenten des VRP kommen kann (Lahyani et al., 2015). Diese Inkompatibilitäten oder auch Abhängigkeiten können über den gesamten Planungshorizont gelten oder auch nur

temporär gültig sein. Dohn et al. (2011) führt das *VRPTW and Temporal Dependencies* (VRPTWTD) ein und beschreibt Lösungsansätze. Mit temporären Abhängigkeiten werden Sequenz- und Synchronisationsrestriktionen beschrieben. Fügenschuh (2006) löst ein solches Problem, bei dem ein öffentlicher Busservice in Abhängigkeit vom Schulbeginn optimiert wird. Die Schulstartzeit ist, wie das Routing der Busse, Lösung der Problemstellung. Fügenschuh (2006) bezeichnet beschriebenes Problem als *VRP with Coupled Time Windows* (VRPCTW). Eine umfangreiche Übersicht über das VRP mit Sequenz- und Synchronisationsrestriktionen ist gegeben in Drexl (2012). Der Autor geht dabei intensiver auf die Synchronisationsrestriktionen ein, welche unter anderem typisch sind für, zum Beispiel, das DARP, oder das VRPPD (eine Person oder Ladung kann erst ausgeliefert werden, wenn diese abgeholt ist). Wie oben erwähnt gibt es nicht nur Abhängigkeiten oder Inkompatibilitäten zwischen einzelnen Ladungen, sondern auch, zum Beispiel, zwischen Fahrzeugen und Gütern, oder Fahrzeugen und Fahrern. Goel und Gruhn (2008) führt das *General VRP* (GVRP) ein, eine Kombination aus VRPTW, HVRP, MDVRP und PDP. Das GVRP berücksichtigt unter anderem auch Kompatibilitätsrestriktionen für Kunden (Aufträge) und Fahrzeuge.

Spezifische Restriktionen

Das VRP ist in unzähligen Domänen anzutreffen und unterliegt somit umfangreichen Anforderungen hinsichtlich der Problemformulierung. Lahyani et al. (2015) kategorisiert Restriktionen, die nur vereinzelt in der Literatur beschrieben werden bzw. sehr domänenspezifisch sind in die Kategorie **Spezifische Restriktionen**. Hierzu gehören, zum Beispiel, Umweltrestriktionen, mit denen sich unter anderem Lin et al. (2014) beschäftigt. Auch die Priorisierung von Kunden (Cornillier et al., 2009), die Reinigung von Fahrzeugen zwischen Transporten (Oppen und Løkketangen, 2006), die Anzahl von Fahrzeugen, die gleichzeitig Kunden bedienen dürfen (Russell et al., 2008), bündelt Lahyani et al. (2015) in dieser Kategorie.

Müssen Fahrer nach dem Bedienen der Kunden (Anfragen) nicht zum Depot zurückkehren spricht man im Allgemeinen vom *Open VRP* (OVRP) eingeführt von Schrage (1981). Das VRPOD (vgl. Kategorie **Fahrzeuge**), ist ein Spezialfall des OVRP. Ein gelegentlich eingesetzter Fahrer muss nicht zum Depot zurückkehren, nachdem alle ihm zugewiesenen Kunden bedient sind. Neben dem OVRP ordnet Lahyani et al. (2015) auch alle VRP die im Kontext von multimodalem Verkehr aufkommen und damit sehr spezifische Restriktionen beschreiben der Klasse **Spezifische Restriktionen** zu. Multimodaler, oder auch intermodaler, oder kombinierter Verkehr ist das Nutzen verschiedener Verkehrsträger in einem bestimmten Zeitraum. Typische Anwendungsfälle sind das Umladen von Gütern zwischen Fahrzeugen oder auch das Nutzen von verschiedenen öffentlichen Verkehrsträgern (Koch, 1997). Unter anderem findet man typische Charakteristika des kombinierten Verkehrs auch im TTRP (vgl. Kategorie **Fahrzeuge**).

Zielfunktion

Die Zielfunktion für das VRP ist oft domänen- und problemspezifisch. In den meisten Fällen wird aber die Fahrzeit, die gefahrene Distanz, die entstandenen Kosten, oder die Fahrzeugflotte minimiert. Maximiert wird häufig die Servicequalität, zum Beispiel, gemessen an der Abweichung vom Zeitfenster, oder auch der durch Bedienen von Kunden entstehende Profit. Üblicherweise verhalten sich die Zielkriterien/Zielwerte gegensätzlich zueinander, sodass nur pareto-optimale Zustände erreicht werden können. In einem pareto-optimalen oder pareto-effizienten Zustand ist es nicht möglich ein Zielkriterium zu verbessern, ohne dabei ein anderes zu verschlechtern (Censor, 1977). Zum Beispiel, ist es in einem solchen Zustand nicht möglich die Servicequalität weiter zu erhöhen, ohne dabei die entstehenden Fahrtkosten zu vergrößern. Lösungsansätze, die für das VRP mehrere Zielkriterien berücksichtigen, sind unter anderem zusammengefasst in Jozefowiez et al. (2008).

Die vorliegende Arbeit beschäftigt sich mit dem DVRP, also mit Routingproblemen, bei denen sich die Probleminformationen über die Zeit ändern. Grundsätzlich kann ein DVRP mit allen anderen Problemeigenschaften aus den oben genannten Kategorien kombiniert werden. In den folgenden Unterkapiteln wird auf die Besonderheiten des dynamischen Routings vertieft eingegangen.

2.1.2 Dynamische Fahrzeugwegeplanung

Das *Dynamic Vehicle Routing Problem* (DVRP) (dt. dynamisches Fahrzeugwegeplanungsproblem) zeichnet sich durch sich ändernden Probleminformationen über die Zeit aus. Eine erste Referenz auf eine solche Problemstellung ist in Wilson und Colvin (1977) zu finden. Beschrieben wird ein computerkontrolliertes Dial-A-Ride System in Rochester, NY (USA). Das System stellt eine dynamische Variante des DARP dar (vgl. Abschnitt 2.1.1). Ein solches System ist eine bedarfsorientierte Art des öffentlichen Personennahverkehrs. Wilson und Colvin (1977) beschreiben, wie kontinuierlich über den Tag verteilte Personentransportaufträge dem Planungssystem bekannt und diese dann automatisiert auf eine Fahrzeugflotte geplant werden. Das *Dynamic TSP* (DTSP), auch ein Spezialfall des DVRP, wird erstmals von Psaraftis (1988) eingeführt. In einem DTSP müssen neben a-priori bekannten statischen Kundenanfragen auch dynamische Kunden durch ein zur Verfügung stehendes Fahrzeug bedient werden. Zusätzlich können in der Beschreibung des DTSP nach Psaraftis (1988) auch Kundenanfragen wieder verschwinden. Die dynamischen Änderungen werden dabei zufällig und unabhängig voneinander einem Planer bekannt. Dynamisch bekannt werdende Kundenanfragen können mithilfe einer zeitabhängigen Kostenmatrix $C(t)$ beschrieben werden. Für einen zum Zeitpunkt t nicht bekannten Kunden v_i sind die Kosten $c_{ji}(t)$ von allen Kunden v_j zu v_i maximal. Dynamisch auftretende Kundenanfragen sind die am häufigsten beschriebene Quelle von Dynamik in einem DVRP (Pillac et al., 2013). Diese dynamischen Kundenanfragen können hinsichtlich von zu transportierenden Gütern (siehe, zum

Beispiel, Attanasio et al. (2004), Hvattum et al. (2006) oder Goel und Gruhn (2008)) oder zu leistenden Services (siehe, zum Beispiel, Bertsimas und Van Ryzin (1991), Bent und Van Hentenryck (2004) oder Beaudry et al. (2010)) unterschieden werden. Weitere Quellen für Dynamik sind dynamische Kundenanfragen für bereits bekannte Kunden (siehe, zum Beispiel, Novoa und Storer (2009) oder Novoa (2005)) und dynamisch zur Verfügung stehende Fahrzeuge (siehe, zum Beispiel, Li et al. (2009*a*) oder Li et al. (2009*b*)). Häufig werden auch Problemstellungen mit schwankenden Transport- oder Servicezeiten als dynamische Routingprobleme beschrieben. Hierbei handelt es sich nach der Problemklassifikation von Lahyani et al. (2015) und nach den Kategorien, hinsichtlich Evolution und Qualität von Probleminformation, von Pillac et al. (2013) aber um stochastische Routingprobleme (vergl. Abschnitt 2.1.1). Zwar können Transport- und Servicezeiten stark schwanken, unterliegen also Unsicherheiten, aber grundsätzlich sind sie einem Planer im Vorfeld bekannt.

Das wesentliche Merkmal von einem dynamischen VRP für die vorliegende Arbeit ist, dass nicht alle Kunden (oder Anfragen) im Vorfeld der Routenplanung bekannt sind. Anfragen werden teilweise erst während der Ausführung der geplanten Routen dem Planer bekannt und werden dann als dynamische Kunden (oder Anfragen) bezeichnet. Während der Ausführung verschwindende Kundenanfragen werden nicht betrachtet. Dynamisch auftretenden Kunden müssen entweder in einen bestehenden Plan integriert werden oder es wird ein neuer Plan generiert. Für jede Anfrage v_i in V' ist zusätzlich ein Zeitpunkt t_i definiert, an dem die Anfrage bekannt wird. Handelt es sich um eine dynamische Anfrage, gilt $t_i > 0$. Für statische Anfragen hingegen ist $t_i = 0$. Abbildung 2.3 verdeutlicht ein DVRP anhand eines einfachen Beispiels. Das linke Bild zeigt den Zustand einer DVRP-Instanz zum Zeitpunkt $t_{DK} > 0$, an dem ein dynamischer Kunde DK bekannt wird. Die Fahrzeuge haben das Depot bereits verlassen und bedienen die ersten statischen Kunden. Die schon gefahrene Strecke ist mit schwarzen Linien dargestellt. Die zu diesem Zeitpunkt geplante Route (gestrichelte Linien) berücksichtigt noch nicht den dynamischen Kunden DK. Im rechten Bild von Abbildung 2.3 ist zu sehen, wie der dynamische Kunde DK, in die geplante Route integriert wird (gestrichelte Linien zu DK). Sind alle statischen und dynamischen Kunden bedient und alle Fahrzeuge zurück im Depot terminiert das Problem.

Anhand des Beispiels aus Abbildung 2.3 wird deutlich, dass die Berechnungszeit $t_{Berechnung}$, die für das Finden einer neuen Route benötigt wird, einen wesentlichen Einfluss hat. Wird zum Beispiel $t_{Berechnung}$ zu groß, kann sich die reale Ausgangslage dahingehend verändert haben, dass die errechnete Lösung nicht mehr umgesetzt werden kann. Auch erhöht sich durch die zusätzliche Komponente Zeit die Problemkomplexität im Vergleich zum statischen VRP (Pillac et al., 2013). Zum einen muss eine zusätzliche Nebenbedingung bei der Lösungsfindung berücksichtigt werden, zum anderen erhöht sich aber auch der Freiheitsgrad bei möglichen umzusetzenden Strategien. So implementiert, zum Beispiel, Bent und Van Hentenryck (2007) Warte- und Platzierungsstrategien um die Zeit zwischen auftretenden dynamischen Kundenanfragen bestmöglich zu nutzen. Auf Warte- und Platzierungsstrategien wird in Abschnitt 2.1.5 genauer eingegangen.

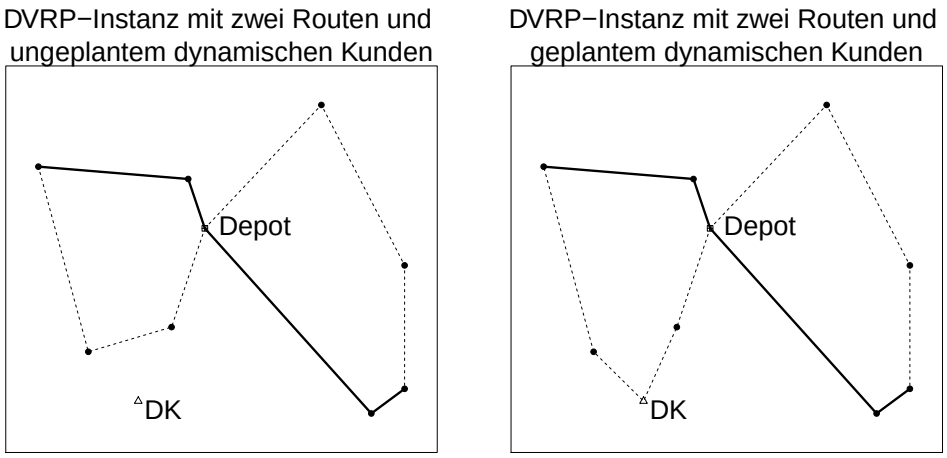

Abbildung 2.3: Visualisierung eines DVRP anhand eines Beispiels. Das linke Bild zeigt die Anfrage des dynamischen Kunden (DK) während die Fahrzeuge das Depot bereits verlassen haben und erste Kunden bedienen. Das rechte Bild zeigt die Integration von DK in eine bestehende Route.

Ein weiterer Unterschied von dynamischem zu statischem Routing ist die bei der Problemlösung betrachtete Kostenfunktion (Psaraftis, 1995). So stehen beim dynamischen Routing meist nicht nur die reinen Transportkosten (Fahrzeiten, zurückgelegte Distanz) im Fokus, vielmehr wird versucht einen Kompromiss zwischen diesen Kosten und den Wartezeiten der Kunden zu erzielen (Bertsimas und Simchi-Levi, 1996). Als Warte- oder auch Antwortzeit wird die Zeit zwischen Bekanntwerden des Kunden (oder der Anfrage) und dem Bedienen dessen bezeichnet. Wenn ein Bedienen der dynamischen Anfragen durch zusätzliche Restriktionen auch abgelehnt werden kann, steht meist auch das Verhältnis zwischen bedienten und abgelehnten dynamischen Kunden im Fokus der Problemlösung. Häufig wird auch der Begriff der Servicegarantie oder des Servicelevels in diesem Zusammenhang genutzt (Hentenryck und Bent, 2009). So versucht, zum Beispiel, der vorgestellte Lösungsansatz in Gendreau et al. (1999) die Anzahl der abgelehnten Anfragen zu minimieren und damit den Durchsatz, also die Anzahl an bedienten Kunden, zu maximieren.

Wesentliche Voraussetzungen für die Anwendung des DVRP in realen Problemstellungen ist die schnelle Berechnung von Problemlösungen, das Erfassen der Standorte der Fahrzeuge und die Kommunikation mit der Flotte in Echtzeit. Nur so können bereits gefahrene Strecken erfasst und geplante Routen kurzfristig neuen Bedarfen angepasst werden. Die Übersichtspapiere von Pillac et al. (2013), Bektaş et al. (2014) und Psaraftis et al. (2016) beschreiben ein stark wachsendes wissenschaftliches Interesse am DVRP ab dem Jahr 2000. Unter anderem macht Psaraftis et al. (2016) auch technologische Fortschritte für dieses gesteigerte Interesse verantwortlich. So kam, zum Beispiel, der Durchbruch des Mobiltelefons auf dem Massenmarkt mit der Veröffentlichung des Nokia 9000 Communicator im Jahr 1996 (Schröder, 2017 und Raczynski und Jörg, 2005). Der Nokia 9000 Communicator ermöglichte neben dem Telefonieren und dem

Schreiben von Kurznachrichten, auch den Zugriff auf Internetseiten mittels Webbrowser. Ab dem Jahr 2000 steht auch das *Global Positioning System* (GPS) ohne künstliche Signalverschlechterung zivilen Nutzern zur Verfügung (Hofmann-Wellenhof et al., 2012). Beide Entwicklungen begünstigen den Einzug von Telemetrie und Telematik in die Transportlogistik (Schreiber, 2004) und bilden damit die Grundlage für das Erfassen der Standorte der Flotte und der Kommunikation in Echtzeit.

Grundsätzlich gilt, dass nicht alle DVRP-Instanzen gleich dynamisch sind (Mayer et al., 2017). Das nachfolgende Kapitel gibt einen Überblick über Ansätze die Dynamik in einer DVRP-Instanz zu charakterisieren.

2.1.3 Erfassen der Dynamik beim Routing

Nach Psaraftis et al. (2016) ist das am häufigsten in der Literatur beschriebene dynamische Element in einer DVRP-Problembeschreibung die Kundenanfrage. Aus diesem Grund konzentriert sich das Erfassen der Dynamik beim Routing auf Kundenanfragen und den Zeitpunkt, wann diese im Verlauf des Problems auftreten. Alle vorgestellten Metriken zum Erfassen der Dynamik in einem DVRP können als Problemeigenschaften interpretiert werden.

Degree of Dynamism

Lund et al. (1996) führt den *Degree of Dynamism* (DOD) als ein Maß für die Dynamik in einem Routingproblem ein. Der DOD ist durch das Verhältnis von dynamischen Kundenanfragen n_{imm} zu der Gesamtzahl der Anfragen n_{tot} definiert (vgl. Gleichung 2.1).

$$DOD = \frac{n_{imm}}{n_{tot}} \tag{2.1}$$

Leicht ist zu sehen, dass in einer rein statischen Problembeschreibung, bei der es keine dynamischen Kundenfragen gibt ($n_{imm} = 0$), der DOD 0 ist. Eine Problembeschreibung, bei der ausschließlich dynamische Kundenanfragen beschrieben werden ($n_{imm} = n_{tot}$), hat einen DOD von 1. Der DOD wird häufig genutzt, um Probleminstanzen voneinander zu unterscheiden (Ritzinger et al., 2016) und die bearbeiteten Instanzen zu charakterisieren. Larsen (2000) führt ein Framework auf Basis des DOD für die Klassifikation von DVRP ein, auf das in Unterabschnitt 2.1.4 genauer eingegangen wird.

Effective Degree of Dynamism

Eine Erweiterung des DOD wird in Larsen (2000) mit dem *Effective Degree of Dynamism* (EDOD) vorgestellt. Der EDOD berücksichtigt neben der Anzahl der dynamischen Kundenfragen auch den Zeitpunkt t_i, an dem die Anfrage v_i in V' dem Planer bekannt wird. Die Berechnungsvorschrift für den EDOD ist in Gleichung 2.2 abgebildet.

$$EDOD = \frac{\sum_{i=1}^{n_{imm}} \frac{t_i}{T}}{n_{tot}} \tag{2.2}$$

T ist dabei definiert als der spät möglichste Zeitpunkt, an dem eine dynamische Kundenanfrage von einem Planer noch verarbeitet werden kann. Das Zeitintervall (0,T] wird üblicherweise als Problem-Zeithorizont bezeichnet. Kilby et al. (1998) beschreibt das Intervall als Konzept für einen Arbeitstag. Häufig wird T in Abhängigkeit von Strukturkomponenten der Problembeschreibung bestimmt. So setzt, zum Beispiel, Kilby et al. (1998) T mit der 8-fachen Distanz zwischen den am weitesten voneinander entfernten Kundenanfragen gleich. Häufig wird T auch mithilfe der entstehenden Lösungskosten unter Berücksichtigung aller Kundenanfragen n_{tot} bestimmt. Genau wie der DOD kann der EDOD nicht kleiner als 0 und nicht größer als 1 werden. Wenn alle Kundenanfragen im System statischer Natur sind, ist der EDOD gleich 0. Bei einem DVRP mit ausschließlich dynamischen Anfragen zum Zeitpunkt T ist der EDOD gleich 1. Zusammenfassend beschreibt Larsen et al. (2007) den EDOD als Repräsentation der zeitlichen Verteilung der dynamischen Kunden v_i. Larsen (2000) evaluiert Ergebnisse von unterschiedlichen Planungsstrategien für das *Partially Dynamic Traveling Repairman Problem* (PDTRP) eingeführt in Bertsimas und Van Ryzin (1991) mithilfe des DOD und des EDOD und kann dabei keinen Mehrwert des EDOD gegenüber des DOD feststellen.

> *..., the EDOD measure does not seem to offer anything special not already captured by the more simple dod measure.*
>
> - Larsen (2000), Seite 75

Effective Degree of Dynamism with Time Windows

Teilweise werden in der Literatur für Kundenanfragen auch zeitliche Restriktionen beschrieben (vgl. Abschnitt 2.1.1). Larsen (2000) erweitert aus diesem Grund den EDOD um eine Reaktionszeit r_i für Anfragen v_i in V'. Die Reaktionszeit definiert die Zeit zwischen Ende des Zeitfensters l_i und dem Zeitpunkt des Auftretens t_i der dynamischen Kundenanfrage. Abbildung 2.4 veranschaulicht den Zusammenhang zwischen Reaktionszeit und Zeitfenster. Die dynamische Kundenanfrage v_1 definiert das Zeitfenster $[e_1, l_1]$ und wird dem Planer zum Zeitpunkt t_1 bekannt. Die Reaktionszeit r_1 definiert die Zeit zwischen Ende des Zeitintervalls l_1 und eintreffen der Anfrage t_1. Die Berechnungsvorschrift für die Reaktionszeit ist in Gleichung 2.3 abgebildet.

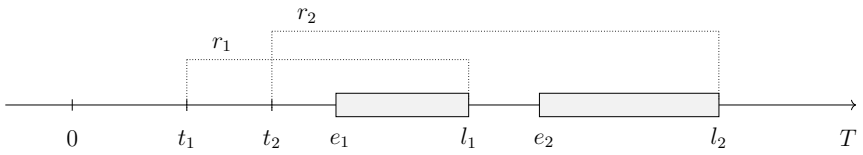

Abbildung 2.4: Veranschaulichung der Reaktionszeit r_i für eine dynamische Kundenanfrage v_i mithilfe eines Beispiels

Der *Effective Degree of Dynamism with Time Windows* (EDOD-TW) berechnet sich nach der Gleichung 2.4. Wie auch für den EDOD erkennt Larsen (2000) keinen

Mehrwert für den EDOD-TW gegenüber dem einfacheren DOD.

$$r_i = l_i - t_i \tag{2.3}$$

$$EDOD\text{-}TW = \frac{1}{n_{tot}} \sum_{i=1}^{n_{tot}} (1 - \frac{r_i}{T}) \tag{2.4}$$

Diskussion

Der DOD hat sich seit seiner Einführung durch Lund et al. (1996) als Maß für die Dynamik in dynamischen Routingproblemstellungen etabliert. Er wird oft herangezogen um bearbeitete Probleminstanzen zu beschreiben und die Performance von Algorithmen zu evaluieren und zu vergleichen (Ritzinger et al., 2016). Der EDOD und auch der EDOD-TW finden aktuell wenig Verwendung, um Probleminstanzen zu charakterisieren und Lösungsansätze zu evaluieren, obwohl das Verhalten von Warte- und Platzierungsstrategien (vgl. Abschnitt 2.1.5) stark von der zeitlichen Verteilung der Kundenanfragen im Zeithorizont beeinflusst wird. Neben der Anzahl von dynamischen Kundenfragen und dem Zeitpunkt der Anfrage gibt es weitere Eigenschaften die häufig in der Literatur für VRP beschrieben werden. Tabelle 2.2 listet typische Eigenschaften von Kundenanfragen. Diese sind teilweise aus dem REP-VRP-Modell, ein von Mendoza et al. (2014) eingeführtes Metamodell für statische VRP, entnommen. Das REP-VRP-Modell wird in Unterabschnitt 2.1.7 im Detail vorgestellt.

Tabelle 2.2: Übersicht über mögliche Eigenschaften einer Kundenanfrage, teilweise Bestandteil des REP-VRP-Modells (Mendoza et al., 2014)

Im REP-VRP-Modell modellierte Eigenschaften von Kundenanfragen	
Knoten	Definiert den Ort (Koordinaten) einer Kundenanfrage
Anzahl	Die Menge, Anzahl oder auch Kapazität der Anfrage
Servicezeit	Die Zeit, die das Durchführen der Anfrage in Anspruch nimmt
Fähigkeit	Benötigte Fähigkeiten zum Durchführen der Anfrage (zum Beispiel: Schulungen, Zertifikate)
Ressourcen	Benötigte Ressourcen zum Durchführen der Anfrage (zum Beispiel: Spezialwerkzeuge)
Dimension	Die Dimension der Anfrage (Länge/Höhe/Breite)
Weitere Eigenschaften von Kundenanfragen	
Gewicht	Das Gewicht der Anfrage

Auch die in Tabelle 2.2 aufgeführten Eigenschaften können in dynamischen Problemstellungen einen großen Einfluss auf das Verhalten von Lösungsansätzen haben und

sollten bei der Bewertung der Dynamik von Probleminstanzen berücksichtigt werden. So weist schon Larsen et al. (2007) darauf hin, dass eine Metrik, die die Dynamik in einem Routingsystem beschreibt, neben der Anzahl der dynamischen Anfragen auch die Servicezeit und die Menge der durch die Kundenanfrage geforderten Güter berücksichtigen sollte. Ein scheinbar wenig dynamisches Routingproblem mit, zum Beispiel, einem DOD von 0.1 bei $n_{tot} = 10$, erweist sich als sehr dynamisch, wenn die Mehrzahl an Gütern durch die einzige dynamische Anfrage transportiert werden muss (siehe Anzahl in Tabelle 2.2), die einzige dynamische Kundenfrage sehr weit von den restlichen Anfragen entfernt ist (siehe Knoten in Tabelle 2.2) oder das einzige vorhandene Spezialwerkzeug (siehe Ressourcen in Tabelle 2.2) benötigt.

Ganz allgemein gilt zu bedenken, dass nur mithilfe des DOD, EDOD und EDOD-TW eine Vielzahl an theoretisch konstruierbaren, unterschiedlichen Probleminstanzen überhaupt nicht voneinander unterschieden werden können. Bei oben genanntem Beispiel mit $n_{tot} = 10$ und einem DOD von 0.1 lassen sich $\binom{n}{k} = \binom{10}{1} = 10$ Probleminstanzen konstruieren. Erscheinen die dynamischen Anfragen in den Probleminstanzen zum gleichen Zeitpunkt, können diese nicht mithilfe der eingeführten Metriken voneinander unterschieden werden. Bei einer Gesamtanzahl von $n_{tot} = 100$ Kundenanfragen und einem DOD von 0.1 sind es schon $\binom{n}{k} = \binom{100}{10} = 17.310.309.456.440$ theoretisch konstruierbare, unterschiedliche Probleminstanzen. 17 Billionen Instanzen die sich aktuell nicht mithilfe vorhandener Metriken voneinander unterscheiden lassen.

Die vorliegende Arbeit leistet hier einen wesentlichen Beitrag um Probleminstanzen besser voneinander unterscheiden zu können. Ausgangspunkt ist dabei die Eigenschaft Knoten (vgl. Tabelle 2.2), eine fundamentale Eigenschaft einer Kundenanfrage, die unabhängig von einer konkreten Problembeschreibung existiert (Mayer et al., 2017). Detailliert werden ortsbezogene Eigenschaften dynamischer Routingprobleme in Kapitel 4 erarbeitet, zusätzlich wird diskutiert, wie diese Eigenschaften zu einer besseren Unterscheidbarkeit von DVRP-Instanzen beitragen.

2.1.4 Problemgruppen und Anwendungsbeispiele

Im folgenden Unterkapitel werden eine Reihe von Anwendungsbeispielen für das DVRP, die in aktuellen wissenschaftlichen Veröffentlichungen diskutiert werden, vorgestellt. Die Auswahl der Veröffentlichungen entspricht einem Auszug aus den 117 Veröffentlichungen, die in dem umfassenden Review „Dynamic Vehicle Routing Problems: Three Decades and Counting“ von Psaraftis et al. (2016) diskutiert werden. Die Anwendungsbeispiele werden in die Klassen **schwach**, **moderat** und **stark dynamische Systeme** unterteilt. Die Klasseneinteilung orientiert sich an Larsen (2000), der ein Framework für die Klassifikation von DVRP anhand des DOD einführt. Larsen (2000) stellt eine Beziehung zwischen Klasse und Zielfunktion her. DVRP mit geringerem DOD (**schwach** bis **moderat dynamische Systeme**) minimieren Routingkosten. Problemstellungen mit größerem DOD (**moderat** bis **stark dynamische Systeme**) hingegen minimieren Antwortzeiten. Zielfunktionen für das VRP und das DVRP werden in Abschnitt 2.1.1 und

Unterabschnitt 2.1.2 diskutiert. Auf die in diesem Abschnitt angeführten Lösungsansätze wird im folgenden Unterabschnitt 2.1.5 im Detail eingegangen.

Das von Larsen (2000) eingeführte Framework für die Klassifikation von Anwendungsfällen definiert keine klaren Grenzen zwischen den Klassen. Für die Zuordnung in dieser Arbeit werden die folgenden Definitionen verwendet. Veröffentlichungen, die sich mit Anwendungsbeispielen beschäftigen, die einen DOD $\leq 0{,}2$ aufweisen, werden der Klasse **schwach dynamische Systeme** zugeordnet. Veröffentlichungen, die der Klasse **moderat dynamische Systeme** zugeordnet werden, beschäftigen sich mit Anwendungen, die einen DOD zwischen 0,2 und 0,8 aufweisen. Publikationen die sich mit Problemen, die einen DOD $\geq 0{,}8$ haben, werden der Klasse **stark dynamische Systeme** zugeordnet. Einer Veröffentlichung können mehrere Klassen zugeordnet sein.

Schwach dynamische Systeme

In **schwach dynamischen Systemen** sind maximal 20% aller Kundenanfragen dynamisch. Klassische Anwendungsbeispiele sind hier Öl- bzw. Gaslieferungen zu privaten Haushalten (Larsen, 2000). Routingprobleme mit nur sehr wenigen dynamischen Anfragen werden häufig auch genutzt um Lösungsmethoden zu evaluieren. So untersucht, zum Beispiel, Branke et al. (2005) Wartestrategien als Lösungsmethode für das DVRP. Branke et al. (2005) führt verschiedene Strategien ein und zeigt empirisch, dass geeignete Wartestrategien den gefahrenen Umweg, um einen dynamischen Kunden zu bedienen, signifikant verringern können. Wartestrategien werden genauer in Abschnitt 2.1.5 vorgestellt. Cheung et al. (2008) entwickelt ein Framework für das dynamische PDP mit Zeitfenstern. Es werden verschiedene Szenarien mit geringem DOD am Beispiel eines Transportunternehmens untersucht.

Moderat dynamische Systeme

Systeme, die sich durch einen DOD größer als 0,2, aber kleiner als 0,8 auszeichnen, bezeichnet Larsen (2000) als **modert dynamische Systeme**. Anwendungsbeispiele für diese Klasse sind, zum Beispiel, lokale Kurier Dienstleister. Fleischmann et al. (2004) entwickelt hierfür ein Planungsframework mit drei eventbasierten Anpassungsalgorithmen unter Berücksichtigung von Online-Verkehrsinformationen. Das Transportieren von Menschen mit Behinderungen oder in hohem Alter ist ebenfalls ein typischer Anwendungsfall für diese Problemklasse. Attanasio et al. (2004) stellt einen parallelen Tabusuchealgorithmus vor, der Lösungen für den Transport von behinderten Menschen erzeugt und optimiert. Eine hybride Tabu-Suche verfolgt auch Berbeglia et al. (2012) für diesen Anwendungsfall. Das Prinzip von Tabusuchealgorithmus wird in Abschnitt 2.1.5 vorgestellt.

Stark dynamische Systeme

In **stark dynamischen Systemen** sind nur wenige Kundenaufträge vor Beginn des Planungshorizonts bekannt. Der in Unterabschnitt 2.1.3 beschriebene DOD ist größer

als 0,8. Die Qualität der Bewirtschaftung solcher hoch dynamischen Systeme wird häufig mithilfe der Reaktionszeit bewertet (Larsen, 2000). Die Reaktionszeit ist die Zeit, die zwischen Anforderung und Ankunft eines Services am dynamischen Kunden vergeht. Oft werden auch Service Level festgeschrieben, die angeben, wie viel Prozent der Aufträge in einer bestimmten Reaktionszeit bedient werden müssen. Für die Lösung stark dynamischer Systeme wird häufig versucht geeignete Warte- und Platzierungsstrategien zu definieren, die teilweise zukünftige Ereignisse implizit oder explizit berücksichtigen (vgl. Unterabschnitt 2.1.5). Warte- und Platzierungsstrategien werden genauer in Abschnitt 2.1.5 vorgestellt. Typische Anwendungsbeispiele für stark dynamische Systeme sind neben Notdiensten wie Polizei, Feuerwehr und ärztliche Dienste (Gendreau et al., 2001 und Beaudry et al., 2010), auch die Lieferung von sehr eiligen Gütern (Ferrucci et al., 2013) oder das Taxigewerbe (Fabri und Recht, 2006).

Diskussion

Abbildung 2.5 visualisiert die Zuordnung aller untersuchten Veröffentlichungen auf die eingeführten Klassen. Gut ist zu sehen, dass sich nur sehr wenige Arbeiten ausschließlich mit schwach dynamischen Systemen auseinandersetzen. Viele Veröffentlichungen betrachten Problemstellungen, die sich nicht eindeutig einer Klasse zuordnen lassen. Dan et al. (2013) untersucht, zum Beispiel, Probleminstanzen mit einem DOD $\in \{0{,}1; 0{,}3; 0{,}5; 0{,}7; 0{,}9\}$ bei der Betrachtung von Notfallmaterialverteilungen. Auch, zum Beispiel, bei der Routenplanung von Transportrobotern in Lagern (Gan et al., 2013) und bei der Planung von Transporten von Gütern zwischen europäischen Flughafen (Goel und Gruhn, 2008) werden Problemstellungen mit einem stark variierenden DOD berücksichtigt.

Grundsätzlich ist aber die Klassifikation von DVRP abhängig vom DOD weit verbreitet und sinnvoll, da sich oft die Zielfunktion von stark dynamischen Systemen im Vergleich zu schwach dynamischen Systemen unterscheidet (Larsen et al., 2008). Trotzdem muss berücksichtigt werden, dass sich hinter einem konkreten DOD eine Vielzahl von verschiedenen Probleminstanzen verbergen (vgl. Unterabschnitt 2.1.3). Eine detaillierte Unterteilung der Klassen erscheint unter Berücksichtigung dieses Aspekts sinnvoll.

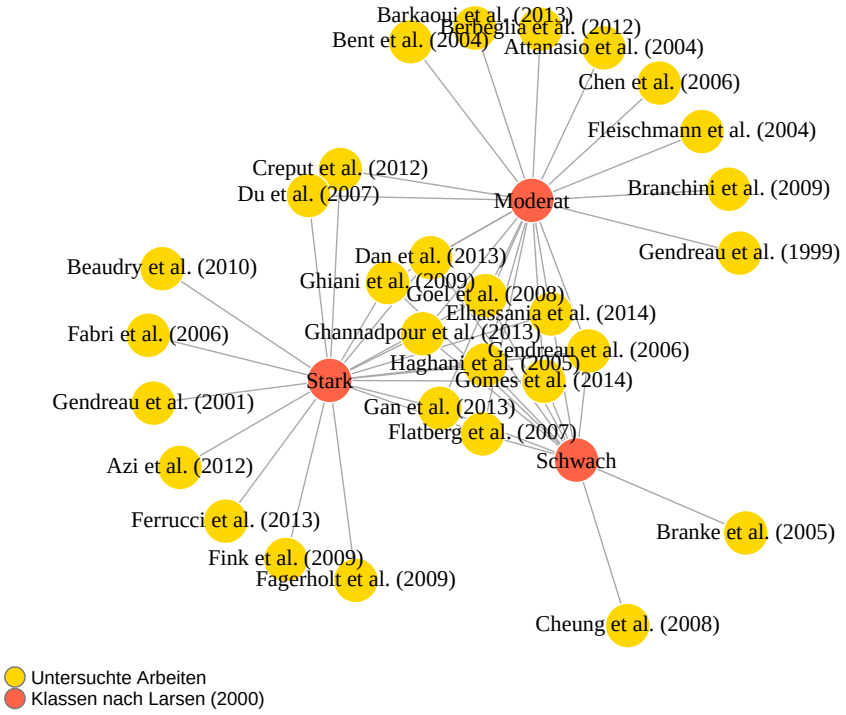

Abbildung 2.5: Zuordnung von untersuchten Arbeiten zu den durch Larsen (2000) eingeführten Klassen

2.1.5 Übersicht über Lösungsansätze

Aktuell eingesetzte Lösungsverfahren für das DVRP basieren überwiegend auf dem Prinzip der rollierenden Planung (Krypczyk, 2010) und werden üblicherweise mithilfe von Simulationen gelöst (Grötschel et al., 2001). Das Prinzip der rollierenden Planung ist in Abbildung 2.6 visualisiert. Zum Zeitpunkt $t_0 = 0$ sind einem Planungsalgorithmus ausschließlich alle statischen Kundenfragen bekannt. Auf der Grundlage dieser Kunden wird ein vorläufiger Routenplan erstellt. Dieser Plan wird bis zum Eintreffen des ersten dynamischen Kunden mithilfe einer Simulation ausgeführt. Werden keine statischen Anfragen in der Problembeschreibung formuliert, entfällt das Planen von initialen Routen. Beim Eintreffen des dynamischen Kunden zum Zeitpunkt $t_1 > t_0$ wird der vorläufige Plan durch einen Anpassungsalgorithmus verändert. Die Simulation führt jetzt den angepassten Plan aus. Dabei können ausschließlich Teilstrecken des Plans angepasst werden, die noch nicht abgefahren worden sind. Der Anpassungsalgorithmus kann gegebenenfalls auch neue Routen, unabhängig von der initialen Planung, erstellen. Der angepasste Plan wird bis zum Eintreffen des nächsten dynamischen Kunden zum Zeitpunkt $t_2 > t_1$ abgearbeitet. Löst jede dynamische Anfrage eine Plananpassung aus, spricht man von einer *Single Event Optimization* (SEO) Strategie (Bianchi, 2000). Die Anpassungen werden hier üblicherweise unregelmäßig, bei bekannt werden eines dynamischen Kunden, ausgelöst. Grundsätzlich muss aber nicht jedes Eintreffen einer dynamischen Kundenanfrage eine Plananpassung auslösen. Eine Anpassung kann

auch regelmäßig stattfinden. Bei dieser Strategie werden dynamische Kundenanfragen gesammelt und dann gemeinsam in den bestehenden Plan integriert.

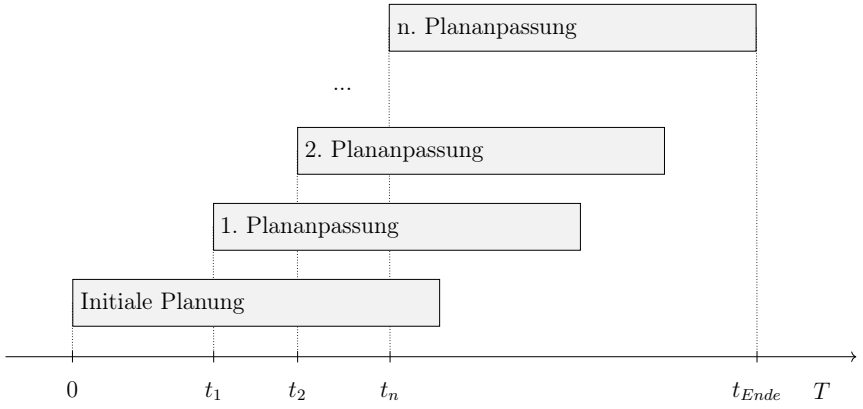

Abbildung 2.6: Visualisierung des Prinzips der rollierenden Planung (Krypczyk, 2010)

Der Planungshorizont T kann begrenzt oder unbegrenzt sein. Bei einem unbegrenzten Planungshorizont wird der Planungsalgorithmus nur einmal ausgeführt. Alle sukzessive bekannt werdenden dynamischen Kundenanfragen werden durch den Anpassungsalgorithmus in den Plan integriert. Üblicherweise ist der Planungshorizont jedoch begrenzt und repräsentiert einen Arbeitstag (vgl. Abschnitt 2.1.3). Alle dynamischen Kunden, die nach $T - \Delta$ eintreffen, werden gesammelt und durch den Planungsalgorithmus als statische Kundenanfragen für den Folgetag berücksichtigt. Verschiedene Problembeschreibung betrachten unterschiedliche Werte für $\Delta = \{x \in \mathbb{R} \mid x \geq 0 \wedge x < T\}$. Durch das Prinzip der rollierenden Planung wird das DVRP in einzelne statische Problemstellungen unterteilt. Dem Planungsalgorithmus stellt sich das DVRP als VRP dar. Der Anpassungsalgorithmus fügt n Anfragen in einen bestehenden Plan ein oder erzeugt einen neuen Plan. Wird ein neuer Plan erzeugt, muss eine Form des OVRP (vgl. Abschnitt 2.1.1) mit dem Depot als letzten zu bedienen Kunden gelöst werden.

Häufig orientieren sich Lösungsalgorithmen an Ansätzen, die sich beim Lösen von statischen Routingproblemen bewährt haben (Psaraftis et al., 2016). Für die Realisierung werden in der Literatur zwei verschiedene Ansätze beschrieben. Ein erster Ansatz ignoriert, die durch den Planungsalgorithmus kreierten Routen und führt eine vollständige Neuplanung mit allen noch planbaren Kundenaufträgen durch. Pankratz (2002) spricht in diesem Fall von einer *„totalen Planrevision“*, die nur für *„voll disponible“* Kunden durchgeführt wird, also für Kunden, die zum Zeitpunkt der Plananpassung noch nicht bedient worden sind. Der zweite Ansatz verändert nur Teile der Planung, erstellt durch den Planungsalgorithmus. Der zweite Ansatz wird in der Literatur als *„partielle Planrevision“* bezeichnet (Pankratz, 2002).

Ein weiteres Unterscheidungskriterium für Lösungsansätze für Planungs- und Anpassungsalgorithmen ist, inwieweit zukünftige Entscheidungen oder auch Vorhersagen über die Entwicklung von dynamischen Kundenanfragen, im Lösungsansatz berücksichtigt werden. Hier unterscheidet Ulmer et al. (2017) die Kategorien Re-Optimierung (RO), *Policy Function Approximation* (PFA), *Lookahead Algorithm* (LA) und *Value Function*

Approximation (VFA). Powell (2011) kategorisiert mithilfe dieser Kategorien Politiken im Kontext der dynamischen Programmierung. Eine Politik ist hier eine Regel oder Funktion, welche eine Entscheidung aus gegebenen Informationen ableitet. Bei kurzsichtigen Politiken (RO) werden Optimierungen unabhängig von vorhergesagten Informationen oder zukünftigen Entscheidungen durchgeführt. Implizit werden zukünftige Entwicklungen in Ansätzen der Kategorie PFA berücksichtigt. In Ansätzen der Kategorien LA und VFA werden zukünftige Entwicklungen und Entscheidungen explizit in die Lösungsfindung integriert. Trotzdem Ansätze, welche Vorhersagen über zukünftige Problementwicklungen berücksichtigen, vielversprechende Lösungen liefern, sind Algorithmen der Kategorie RO weiterhin stark verbreitet (Ulmer et al., 2017).

Methodisch werden Planungs- und Anpassungsalgorithmen mit den unterschiedlichsten heuristischen und metaheuristischen Verfahren adressiert. Psaraftis et al. (2016) identifiziert hier Lösungsklassen, die im Folgenden auszugsweise aufgezählt und kurz erläutert werden. Zum besseren Verständnis der erläuterten Verfahren wird kurz der Begriff Nachbarschaft im Kontext eines Optimierungsproblems erläutert.

Nachbarschaft

Wenn I eine Instanz eines Optimierungsproblems ist, ist $S(I)$ die Menge aller möglichen Lösungen für I. Für jede Lösung $s \in S(I)$ kann eine Nachbarschaft $N(s) \subseteq S(I)$ definiert werden. Die Nachbarschaft $N(s)$ ist also eine Menge von Lösungen in $S(I)$. Wie die Nachbarn $n \in N(s)$ aus $S(I)$ ausgewählt werden ist abhängig von N. Häufig kommen Operatoren zum Einsatz, die durch Vertauschungen oder durch Entfernen und Hinzufügen eine Nachbarschaft $N(s)$ aus s erzeugen. Die Größe der Nachbarschaft kann beliebig gewählt werden, sie bestimmt maßgeblich den Aufwand bei der Erstellung der Nachbarschaft. Die größtmögliche Nachbarschaft von s ist $S(I) \setminus s$. Werden n und s als Nachbarn bezeichnet, bedeutet das, dass n und s sich sehr ähnlich sind. Im Suchraum sind diese Lösungen also sehr nah beieinander und zeigen geringe Werte als Ergebnisse bei der Anwendung eines beliebigen Distanzmaßes.

Tabusuche, parallele und hybride Tabusuche

Die Tabusuche ist ein metaheuristisches Suchverfahren, eingeführt von Glover (1986), bei der iterativ eine Ausgangslösung verbessert wird. Bei der klassischen Tabusuche wird in einer Iteration ausschließlich eine beste Lösung betrachtet. Diese Lösung ist Ausgangspunkt für die Generierung von Nachbarlösungen mithilfe von Operatoren. Die beste Nachbarlösung wird für die weiterführende Suche ausgewählt. Die Tabusuche charakterisiert sich durch die Aktualisierung einer Tabuliste in jeder Iteration. Abhängig von einer Tabustrategie werden Operatoren für eine definierte Anzahl von Iteration tabuisiert. So kann, zum Beispiel, verhindert werden, dass komplementäre Operationen in aufeinanderfolgenden Iterationen ausgeführt werden. Die Grundidee der Tabusuche ist die Steuerung einer lokalen Suche (Operatoren und Auswahl der Lösungen) unter Verwendung einer Gedächtnisstruktur (Tabuliste). Die Tabusuche

wird seit ihrer Einführung kontinuierlich weiterentwickelt. So hat sich, zum Beispiel, die parallele Tabusuche etabliert, bei der in jeder Iteration mehrere Lösungen parallel verbessert werden. Gendreau et al. (1999) implementieren eine parallele Tabusuche zum Verteilen von Zustellungsaufträgen mit Zeitfenstern auf eine Flotte von Fahrzeugen eines Kurierdienstleisters. Attanasio et al. (2004) vergleichen verschiedene Implementierungen von parallelen Tabusuchen für das dynamische DARP. Die Autoren zeigen, dass die parallele Tabusuche gute Ergebnisse für die genannten Anwendungsfälle liefert. Auch wird die Tabusuche oft mit weiteren metaheuristischen Suchverfahren zu hybriden Suchen kombiniert.

Variable Nachbarschaftssuchen, adaptive und große Nachbarschaftssuchen

Die Variable Nachbarschaftssuche, eingeführt von Mladenović und Hansen (1997), ist eine metaheuristische Suche, die eine lokale Suche steuert. Ausgehend von einer Startlösung werden iterativ größer werdende Nachbarschaften mithilfe von Operatoren erzeugt und nach besseren Lösungen durchsucht. Wird eine bessere Lösung gefunden, bildet diese die Grundlage für die Generierung von neuen, größer werdenden Nachbarschaften für die weitere Suche. Gendreau et al. (2006) implementieren eine Nachbarschaftssuche für ein dynamisches PDP. Zum Lösen von DVRP kommen verschiedene Ansätze von Nachbarschaftssuchen zum Einsatz. Azi et al. (2012) entwickeln, zum Beispiel, eine adaptive große Nachbarschaftssuche für ein DVRP. Für eine große Nachbarschaftssuche werden Nachbarn mithilfe von Entferne- und Hinzufügeoperatoren generiert. Eine adaptive große Nachbarschaftssuche verfügt über eine Vielzahl solcher Operatoren und kann diese miteinander kombinieren.

Einfügemethoden

Einfügemethoden finden häufig Anwendung bei der Konstruktion von Lösungen für ein Routingproblem. Ausgehend von einem Startknoten werden sukzessive alle verbleibenden Knoten in eine Lösung integriert, bis keine verbleibenden Knoten mehr existieren und eine valide Lösung erzeugt ist. Eine Einfügemethode muss mindestens drei Entscheidungen treffen. Die erste Entscheidung betrifft die Auswahl des Startknotens. Die zweite Entscheidung beschäftigt sich mit der Auswahl des Knotens, der als nächster der existierenden Teillösung hinzugefügt werden soll. Die dritte Entscheidung legt die Position fest, an der der ausgewählte Knoten eingefügt wird. Einfügemethoden kommen in einem dynamischen Problem häufig zur Anwendung, um die dynamisch auftretenden Knoten in die bestehende Lösungen zu integrieren. Beaudry et al. (2010), zum Beispiel, verwenden eine Einfügemethode, um einen Patiententransportauftrag in eine bestehende Lösung zu integrieren. Der Auftrag unterliegt dabei einer Vielzahl von Restriktionen, sodass die Konstruktion von validen Lösungen eine erhebliche Herausforderung darstellt. Die entstehende Lösung wird durch eine nachgelagerte Tabusuche verbessert.

Ameisenalgorithmen

Ameisenalgorithmen (*Ant System* (AS)), erstmalig eingeführt in Colorni et al. (1991), sind populationsbasierte Metaheuristiken, bei der durch künstliche Ameisen Lösungen für Optimierungsprobleme konstruiert werden. Dorigo (1992) stellt einen Ameisenkolonie-Algorithmus (*Ant Colony Optimization* (ACO)) zum Lösen von VRP vor. Jede Ameise erzeugt pro Generation eine Lösung für das gegebene Problem. Bei der Konstruktion von Lösungen wird auf heuristische Informationen und auf Erfahrungen, gesammelt durch die gesamte Ameisenkolonie zurückgegriffen. Erfahrungen der Kolonie werden mithilfe einer Pheromonmatrix kommuniziert. Nachdem jede Ameise der Kolonie eine Lösung in einer Generation erzeugt hat, wird die Lösung bewertet und die Matrix angepasst. Gute Entscheidungen auf dem Weg zur Lösungsfindung werden dabei verstärkt. Bevor eine neue Generation von Ameisen weitere Lösungen erzeugt, werden die restlichen Pheromonwerte reduziert. Viele Weiterentwicklungen des Algorithmus beschäftigen sich, zum Beispiel, mit dem Update der Pheromonwerte. Mavrovouniotis et al. (2015), zum Beispiel, verwenden ein *Max Min AS* (MMAS), um ein DTSP zu lösen. In einem MMAS wird eine Ober- und Untergrenze für die Pheromonwerte definiert. Auch Dan et al. (2013) nutzen ein AS, um Lösungen für den Transport von Notfallmaterialien zu erzeugen.

Warte- und Platzierungsstrategien

Bei Warte- und Platzierungsstrategien handelt es sich um Lösungsansätze, die ihren Ursprung nicht in der Lösung von statischen Routingproblemen haben. Eine Wartestrategie zeichnet sich durch die verzögerte Zuweisung von dynamischen Anfrage an Fahrzeuge aus. Während das Fahrzeug wartet werden dynamische Anfragen gesammelt und anschließend gemeinsam zu einer Route für das Fahrzeug geplant. Die Flexibilität des Systems für mögliche weitere dynamische Anfragen soll dadurch maximiert werden. Eine Platzierungsstrategie beschreibt das Verschieben von unbeschäftigten Fahrzeugen in Gebiete, für die zukünftige, dynamische Anfragen erwartet werden. Oft werden beide Strategien miteinander kombiniert und in stark dynamischen Systemen eingesetzt. Pureza und Laporte (2008) untersuchen die Auswirkung von Wartestrategien bei der Planung einer dynamischen Anfrage auf ein konkretes Fahrzeug. Auch wird untersucht, inwieweit das Warten zwischen dem Bedienen von Anfragen einen Einfluss auf die Lösungsqualität hat. Ein weiterer Ansatz ist das Warten in geeigneten Gebieten gezielt in eine Route einzuplanen, um die Reaktionszeit auf mögliche dynamische Ereignisse zu verkleinern. Diese Idee verfolgt unter anderem die Arbeit von Branke et al. (2005). Toregas et al. (1971) hingegen beschäftigt sich mit der Platzierung von Notfalleinrichtungen, mit dem Ziel der Minimierung benötigter Krankenwagen. Brotcorne et al. (2003) gibt einen Überblick über Platzierungsmodelle von Krankenwagen, entwickelt seit 1970. Gendreau et al. (2001) und Ghiani et al. (2008) entwerfen Platzierungsstrategien zum Verkürzen der Reaktionszeit. Die Hauptidee ist dabei das Verschieben von unbeschäftigten Fahrzeugen in Gebiete mit einer hohen Wahrscheinlichkeit für zukünftige Anfragen.

Unter anderem weisen auch Bent und Van Hentenryck (2007) den Nutzen von a priori Informationen im Hinblick auf die Lösungsqualität durch das Einführen von Warte- und Platzierungsstrategien nach.

Diskussion

Neben den aufgezählten Klassen identifiziert Psaraftis et al. (2016) weitere Lösungsklassen. So werden DVRP häufig auch mit den naturanalogen Optimierungsverfahren Partikelschwarmoptimierung und Evolutionärer Algorithmus (EA) gelöst. Werden neben dynamischen Kundenanfragen auch stochastische Aspekte im DVRP betrachtet, wird das Problem auch mithilfe von Markow-Entscheidungsprozessen modelliert.

Abbildung 2.7 zeigt eine Zuordnung von den durch Psaraftis et al. (2016) identifizierten Lösungsklassen auf die in dieser Arbeit betrachteten wissenschaftlichen Veröffentlichungen. Die Veröffentlichungen sind den durch Larsen (2000) eingeführten Klassen (vgl. Unterabschnitt 2.1.4) zugeordnet. Gut ist zu sehen, dass mehrfach verwendete Lösungsansätze nicht eindeutig bestimmten Gruppe zugeordnet werden können. Das bedeutet, dass die Lösungsverfahren unabhängig von der beschreibbaren Dynamik in DVRP zum Einsatz kommen und eine Wahl für einen Algorithmus allein anhand des DOD nicht möglich ist. Es werden zusätzliche Problemeigenschaften benötigt, mit denen Probleminstanzen voneinander unterschieden werden können, um Algorithmen besser vergleichen und gezielt für Instanzen auswählen zu können.

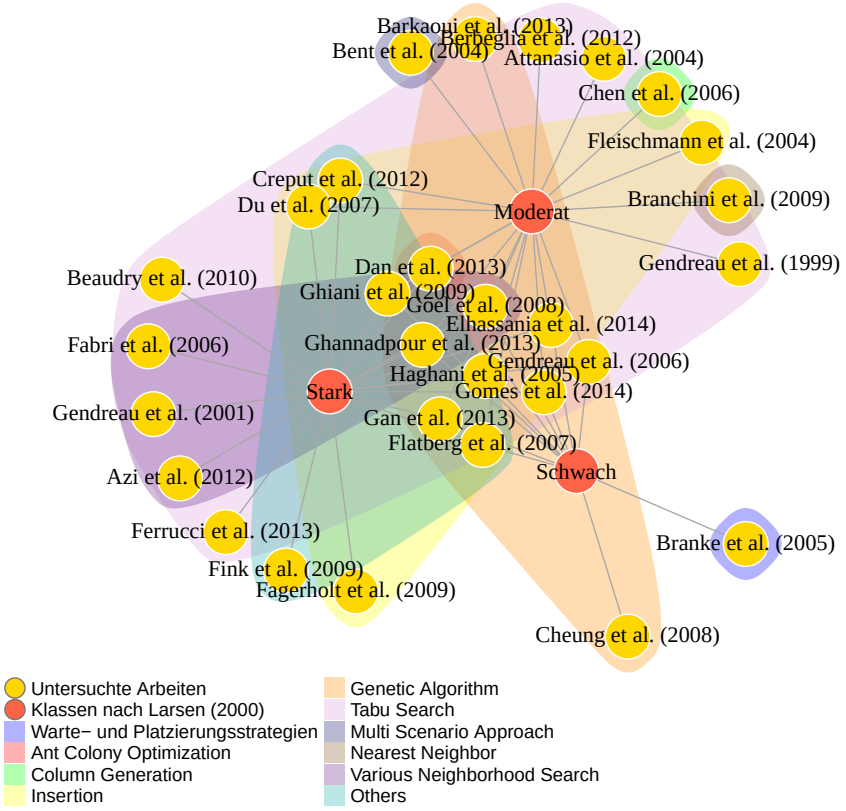

Abbildung 2.7: Zuordnung von untersuchten Arbeiten zu den durch Larsen (2000) eingeführten Klassen, mit farblicher Zuordnung zu verwendeten Lösungsansätzen

2.1.6 Bewerten der Qualität von Lösungen

Die Bewertung der Lösungsqualität von statischen Optimierungsproblemen gestaltet sich einfach. Häufig werden die Laufzeit oder das Ergebnis der Zielfunktion (Lösungskosten) betrachtet (Pillac et al., 2013). Dynamische Probleme hingegen erfordern das Einführen neuer Metriken, um Lösungen eines Algorithmus zu bewerten. Sleator und Tarjan (1985) führen für die Bewertung der Lösungsqualität eines dynamischen Optimierungsproblems die *Competitive Analysis* ein. Betrachtet wird die Menge F von Probleminstanzen eines dynamischen Minimierungsproblems P. Dabei sind $z^*(I_{statisch})$ die optimalen Lösungskosten für die statische Probleminstanz $I_{statisch}$ der dynamischen Instanz $I \in F$. Die statische Probleminstanz $I_{statisch}$ zeichnet sich dadurch aus, dass alle Probleminformationen, also auch die dynamischen, bei der Konstruktion der initialen Lösung zur Verfügung stehen. Eine dynamische Lösungsanpassung ist für $I_{statisch}$ nicht erforderlich. Bei der dynamischen Instanz hingegen werden Probleminformationen erst nach der Konstruktion der initialen Lösung, während der Ausführung der Lösung für das Problem, bekannt (vgl. Unterabschnitt 2.1.5). Ein Algorithmus A, der die Lösungskosten $z_A(I)$ für die dynamische Probleminstanz I generiert, wird *c-competitive* genannt, wenn

ein α existiert, sodass die Ungleichung 2.5 erfüllt ist.

$$z_A(I) \leq c * z^*(I_{statisch}) + \alpha, \forall I \in F \tag{2.5}$$

Ist $\alpha = 0$, handelt es sich um einen *streng c-competitive* Algorithmus. Hier ist das Ergebnis der Zielfunktion der durch A erzeugten Lösung immer c-mal größer als der optimale Wert. Weiterführende Informationen über die *Competitive Analysis* sind, zum Beispiel, in Borodin und El-Yaniv (2005) zu finden.

Der Hauptnachteil der *Competitive Analysis* ist, dass die eingeführte Ungleichung für jeden untersuchten Algorithmus analytisch nachgewiesen werden muss (Pillac et al., 2013). Der Nachweis kann für große Problemstellungen äußerst komplex sein (Grötschel et al., 2001). Das von Mitrović-Minić et al. (2004) eingeführte *Value of Information* hingegen, bewertet die Lösungsqualität von Algorithmen basierend auf empirischen Ergebnissen. Durch die Bestimmung des *Value of Information* V_A, können Algorithmen hinsichtlich ihres Umgangs mit der Dynamik in einer Problemstellung bewertet werden (Cheung et al., 2008). $V_A(I)$, für eine Probleminstanz I, für einen Algorithmus A, berechnet sich mithilfe der Lösungskosten $z_A(I)$ erzeugt durch einen Algorithmus A für I und den Lösungskosten $z_A(I_{statisch})$ für das statische Problem $I_{statisch}$. Die Gleichung für die Berechnung des *Value of Information* nach Pillac et al. (2013) ist in 2.6 dargestellt.

$$V_A(I) = \frac{z_A(I) - z_A(I_{statisch})}{z_A(I_{statisch})} \tag{2.6}$$

Mithilfe des *Value of Information* kann geschlussfolgert werden, wie gut ein Algorithmus auf dynamische Änderungen reagieren kann. Ein Vergleich von verschiedenen Algorithmen mithilfe des *Value of Information* gestaltet sich hingegen schwierig. Es lässt sich keine Aussage über die generelle Qualität einer Lösung in Bezug auf die Probleminstanz ableiten. Kann gezeigt werden, dass $V_{A_1}(I) < V_{A_2}(I)$ gilt, bedeutet das lediglich, dass Algorithmus A_1 mit der Dynamik in I besser umgehen kann als A_2. Die Qualität der Lösung von A_2, kann dennoch besser sein, als die Qualität der Lösung von A_1 ($z_{A_1}(I) > z_{A_2}(I)$). Aus diesem Grund wird, für den Vergleich von Algorithmen, das *Value of Information* für diese Arbeit erweitert. Die Erweiterung $V_A^*(I)$ betrachtet neben den Lösungskosten $z_A(I)$ des zu untersuchenden Algorithmus A für die dynamische Probleminstanz I auch die Lösungskosten z_{Opt^*} eines nahe optimalen Algorithmus Opt^* für die statische Probleminstanz $I_{statisch}$. Die Berechnung der Erweiterung des *Value of Information* ist in Gleichung 2.7 abgebildet.

$$V_A^*(I) = \frac{z_A(I) - z_{Opt^*}(I_{statisch})}{z_{Opt^*}(I_{statisch})} \tag{2.7}$$

Mithilfe des erweiterten *Value of Information* lässt sich die Lösungsqualität von Algorithmen auch untereinander vergleichen, weil die Vergleichsbasis keine durch den Algorithmus A erzeugte Lösung ist, sondern eine Näherungslösung erzeugt durch Opt^*. Das kann unter anderem dazu führen, dass $V_A^*(I) > 0$ ist, selbst wenn $z_A(I) = z_A(I_{statisch})$

gilt. Der lineare Verlauf des erweiterten *Value of Information* ist in Abbildung 2.8 abgebildet. Gut ist zu sehen, dass $V_A^*(I) = 0$ gilt, wenn das Ergebnis des Algorithmus A für die dynamische Problemstellung gleich ($\beta = 1$) der Lösung des Algorithmus Opt^* für die statische Lösung $I_{statisch}$ ist ($z_A(I) = z_{Opt^*}(I_{statisch})$). Ist die Lösung von A doppelt so groß ($\beta = 2$) wie die Lösung von Opt^* ist $V_A^*(I) = 1$.

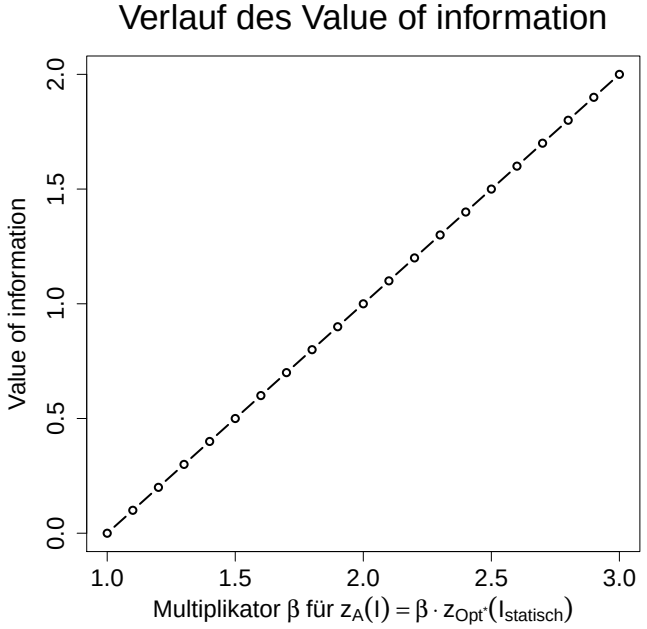

Abbildung 2.8: Der Verlauf des erweiterten *Value of Information* in Bezug zu dem Multiplikator β aus $z_A(I) = \beta * z_{Opt^*}(I_{statisch})$

Auswertungen und Vergleiche von Algorithmen in dieser Arbeit basieren auf dem erweiterten *Value of Information*. So ist gewährleistet, dass die Algorithmen in den Darstellungen miteinander verglichen werden können und auch ein Ergebnis eines Algorithmus in Relation zu der nahe optimalen Lösung der statischen Problemstellung interpretiert werden kann.

2.1.7 Benchmarkproblemstellungen

Für das DVRP stehen kaum Benchmarkproblemstellungen zur Verfügung (Pillac et al., 2013). Orientieren sich Veröffentlichungen nicht an realen Problemstellungen, werden häufig, für die Untersuchung von Algorithmen künstliche Probleminstanzen zufällig erzeugt. Teilweise werden auch aus statischen Instanzen dynamische generiert. Hentenryck und Bent (2009), zum Beispiel, beschreiben, wie aus statischen Benchmarks dynamische Probleminstanzen generiert werden können. Die Arbeit basiert auf Instanzen eingeführt von Solomon (1987). Auch Kilby et al. (1998) beschreiben ein Vorgehen für die Transformation von statischen zu dynamischen Instanzen. Kilby et al. (1998) nutzt statische Probleminstanzen eingeführt von Christofides und Beasley (1984). Für das

dynamische TSP stellen Mavrovouniotis et al. (2012) einen Benchmarkgenerator (BG) bereit, der aus beliebigen statischen TSP-Instanzen dynamische generieren kann.

Für statische Routingproblemstellungen hingegen finden sich viele Benchmarkprobleme. Das VRP-Repository (Mendoza et al., 2019), zum Beispiel, bietet eine sehr umfangreiche Sammlung von statischen Probleminstanzen. Das Repository erlaubt die Kommunikation von Instanzen auf der Basis eines VRP Metamodells (REP-VRP-Modell), eingeführt von Mendoza et al. (2014). Mayer et al. (2016) klassifizieren das REP-VRP-Modell nach der aktuellen RVRP-Klassifikation eingeführt von Lahyani et al. (2015). Es zeigt sich, dass das Metamodell eine Vielzahl von verschiedenen statischen Variationen des VRP unterstützt. Im Zuge dieser Arbeit wird eine Erweiterung des REP-VRP-Modells erarbeitet, die zusätzlich eine Abbildung von dynamischen Problemstellungen erlaubt (vgl. Abschnitt 3.3). Sim et al. (2015) veröffentlichen ebenfalls ein Metamodell für die Modellierung von VRP. Das „XML Object Model für Rich Vehicle Routing Problems“ unterstützt weniger Problemvariationen und modelliert Teilaspekte mithilfe von Enumerationen, was die Flexibilität des Modells stark einschränkt (vgl. Mayer et al., 2016).

2.2 Das Algorithm Selection Problem

Das Problem der Wahl eines Algorithmus, der eine bestimmte Problemstellung besonders effektiv oder gut löst, ergibt sich in unzähligen Situationen. Nach wie vor gilt, dass Optimierungsprobleme in NP mit heuristischen Ansätzen gelöst werden müssen. Eine vollständige Enumeration über alle möglichen Lösungen, um das Optimum zu finden, ist trotz modernster Computerhardware nicht möglich (Russell und Norvig, 2012). Ein Optimierungsproblem liegt in NP, wenn nur ein nichtdeterministischer Algorithmus, der das Problem in Polynomialzeit löst, gefunden werden kann (Wegener, 2013). Wolpert und Macready (1997) führen die No-Free-Lunch-Theoreme ein. Die Theoreme zeigen, dass die durchschnittliche Lösungsqualität, erzeugt durch unterschiedliche heuristische Algorithmen, über alle möglichen Problemstellungen identisch ist. Kann also eine bessere Performance für einen Algorithmus A für eine Gruppe von Problemstellungen im Vergleich zu einem anderen Algorithmus B nachgewiesen werden, wissen wir, dass Problemstellungen existieren müssen, bei denen B erfolgreicher ist als A. So ergibt sich aus den Theoremen, eingeführt von Wolpert und Macready (1997), dass es keine Heuristik gibt, die allen anderen für alle Problemstellungen überlegen ist.

In Online-Optimierungsproblemen wie dem DVRP müssen Entscheidungen während des Ausrollens der geplanten Routen durch Algorithmen getroffen werden. Die getroffenen Entscheidungen wirken sich direkt in der Realität aus und ein Revidieren ist nicht möglich. Um so wichtiger ist aus diesem Grund die Auswahl eines Algorithmus basierend auf Problemeigenschaften. Bei statischen Problemstellungen werden alle Entscheidungen durch einen Algorithmus offline getroffen. So können verschiedene Algorithmen erprobt und miteinander vergleichen werden, noch bevor Lösungen in der Realität ausgerollt werden. Bei Online-Optimierungsproblemen ist das nicht möglich.

Im folgenden Unterabschnitt 2.2.1 wird das *Algorithm Selection Problem* (ASP) eingeführt, die Formalisierung des Problems der Wahl eines Algorithmus auf der Grundlage von Problemeigenschaften und der Performance von Algorithmen. In Unterabschnitt 2.2.2 werden problemabhänge und unabhängige Eigenschaften diskutiert und Eigenschaften für die Auswahl von Algorithmen im Kontext des TSP vorgestellt. Der dann nachfolgende Unterabschnitt setzt sich kurz mit üblichen Methoden der Exploration des Performanceraums auseinander. Unterabschnitt 2.2.4 beschäftigt sich mit den verschiedenen Ansätzen für die Selektion von Algorithmen. Es werden kurz künstliche neuronale Netze erläutert, die in dieser Arbeit für die Modellierung der Beziehung zwischen Problemeigenschaften und Performance verwendet werden. Abschließend werden Anwendungsfälle diskutiert und Forschungen vorgestellt, die sich mit dem ASP in der Domäne der dynamischen Fahrzeugwegeplanung auseinandersetzen.

2.2.1 Problembeschreibung

Rice (1976) formalisiert das ASP und schlägt ein Framework vor, mit dessen Hilfe vorhergesagt werden soll, welcher Algorithmus aus einer Menge von Algorithmen für eine Probleminstanz die beste Performance verspricht. Die Auswahl eines Algorithmus soll dabei anhand von messbaren Eigenschaften der Problemstellung erfolgen (Smith-Miles und Lopes, 2012). Grundsätzlich ist eine Auswahl von Algorithmen auch auf der Grundlage der vollständigen Probleminstanz selbst möglich. Eine Reduktion der Instanz auf wichtige Eigenschaften vereinfacht die Modellbildung für die Auswahl von Algorithmen jedoch signifikant. Häufig enthält die Repräsentation einer Probleminstanz Informationen, die keinen Einfluss auf die Performance von Algorithmen haben. Diese Informationen werden durch das Erfassen von relevanten Problemeigenschaften im Vorfeld der Algorithmenseletion gefiltert und müssen nicht aufwendig durch ein Selektionsmodell identifiziert werden. Die Größe der Datenbasis korreliert mit dem Aufwand der Suche nach einem erfolgreichen Selektionsmodell und wird aus diesem Grund auf die wesentlichen Informationen beschränkt. Für das in dieser Arbeit betrachtete DVRP ist eine Selektion von Algorithmen auf der Grundlage der vollständigen Probleminstanz zudem ungeeignet, da die Algorithmenauswahl auf Prognosen über die DVRP-Instanz basiert. Durch eine auf Eigenschaften basierte Auswahl von Algorithmen kann eine Prognose aller dynamischer Informationen vermieden werden. Ausschließlich für die wichtigsten Problemeigenschaften müssen so Prognosemodelle erstellt werden. Abbildung 2.9 zeigt das Framework für das ASP. In der folgenden Aufzählung werden die vier Hauptkomponenten des Frameworks kurz erläutert.

- Der Problemraum P repräsentiert die Menge aller Problemstellungen.
- Der Eigenschaftenraum F beinhaltet alle an einer Problemstellungen x messbaren Eigenschaften.
- Der Algorithmenraum A bündelt die Algorithmen, die Probleme aus F lösen können.

- Der Performanceraum beschreibt die Zuordnung von dem Ergebnis eines Algorithmus zu einer Performance-Metrik, zum Beispiel, Lösungskosten oder Laufzeit.

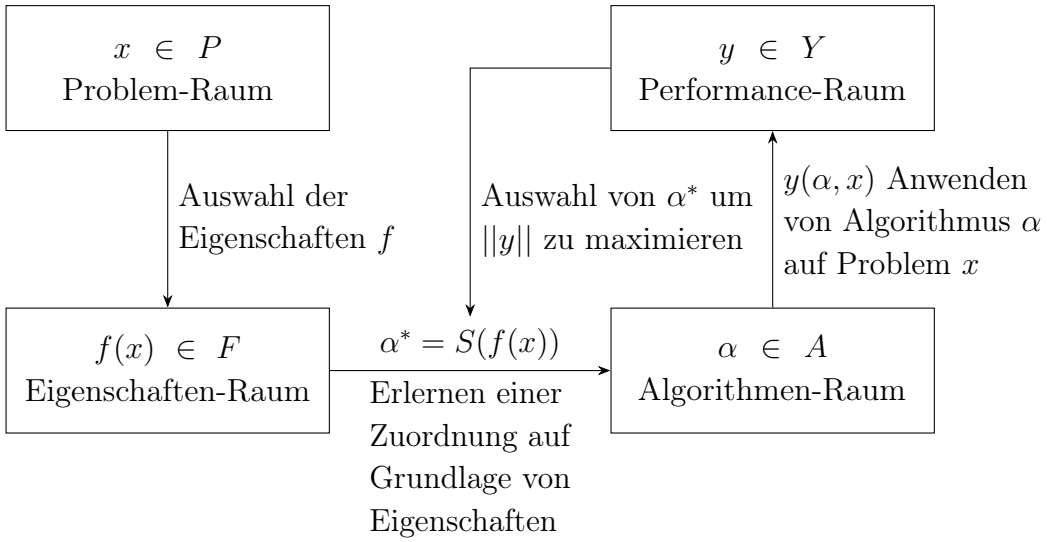

Abbildung 2.9: Das von Rice (1976) eingeführte Framework für das ASP

Das ASP Framework versucht für eine Probleminstanz $x \in P$, mit den Eigenschaften $f(x) \in F$, eine Abbildung $S(f(x))$ in den Algorithmenraum A zu finden, sodass der selektierte Algorithmus $\alpha^* \in A$ die Performancemetrik $||y||$ für $y(\alpha,x) \in Y$ maximiert (Smith-Miles, 2009).

Die Grundidee hinter dem Framework ist die Vorhersage der Performance eines Algorithmus für eine Probleminstanz anhand messbarer Eigenschaften. Viele wissenschaftliche Arbeiten beschäftigen sich mit der Vorhersage der Performance auch im Kontext der Erörterung der Schwere von Optimierungsproblemen. Dabei wird der Fragestellung nachgegangen, ob bestimmte Ausprägungen von Eigenschaften auf besonders schwer zu lösende Probleme hindeuten. Die Schwere einer Problemstellung muss dabei immer in Abhängigkeit eines bestimmten Lösungsalgorithmus betrachtet werden (Macready und Wolpert, 1996). Smith-Miles und Lopes (2012) identifizieren drei Hauptströmungen die Algorithmenselektion oder Untersuchungen hinsichtlich der Schwere von Optimierungsproblemen adressieren. Studien aus dem Bereich der künstlichen Intelligenz konzentrieren sich oftmals auf das *Satisfiability Problem* (SAT). Die Lösung der Problemstellung verlangt das Finden von einem Zustand, der alle formulierten Bedingungen erfüllt. Die Eigenschaften der Problemstellung werden genutzt, um die Laufzeit von mathematischen Solvern oder Heuristiken vorherzusagen. Aktuelle Ansätze aus dem Bereich fokussieren sich auf die dynamische Auswahl von Teilen von Algorithmen in Echtzeit, siehe, zum Beispiel, die Veröffentlichungen von Streeter et al. (2007) oder Samulowitz und Memisevic (2007). Im Bereich des maschinellen Lernens wird das Meta-Lernen fokussiert, also das Lernen über das Lernen, für Klassifikationsprobleme (Smith-Miles und Lopes, 2012). Methoden des überwachten Lernens werden eingesetzt, um den Zusammenhang zwischen den Eigenschaften einer Problemstellung

und dem Ergebnis einer Klassifikation zu ergründen. Dabei werden typische Klassifikatoren wie Entscheidungsbäume, neuronale Netze oder *Support Vector Machines* (SVM) (dt. Stützvektormaschinen) untersucht. Vilalta und Drissi (2002) geben einen detaillierten Überblick über das Meta-Lernen. Die dritte, von Smith-Miles und Lopes (2012) identifizierte Strömung hat ihren Ursprung im Operations Research. Häufig steht hier die Analyse der Eigenschaften des Suchraums im Vordergrund. Bierwirth et al. (2004) untersucht, zum Beispiel, die Fitnesslandschaft des *Job Shop Scheduling Problems*. Die Autoren empfehlen die Berücksichtigung der Beschaffenheit der Fitnesslandschaft bei der Konstruktion bzw. Auswahl von Algorithmen. Hyper-Heuristiken nutzen ebenfalls die Beschaffenheit der Fitnesslandschaft. Eine übergeordnete Heuristik wählt dynamisch eine passende, einfachere Heuristik anhand ihrer Fitness (Burke et al., 2003). Die Performance der einfacheren Heuristik kann als Eigenschaft im Kontext des von Rice (1976) eingeführten Frameworks interpretiert werden (Smith-Miles und Lopes, 2012).

Einen sehr detaillierten Überblick über Algorithmenselektion in den verschiedenen Bereichen bietet neben Smith-Miles und Lopes (2012) auch unter anderem Smith-Miles (2009).

2.2.2 Eigenschaftenraum

Anhand der aufgeführten Beispiele für Algorithmenselektion können zwei grundsätzliche Unterscheidungen bei der Betrachtung des Eigenschaftenraums nachvollzogen werden. Der Raum kann entweder unabhängig oder abhängig von der konkreten Problemstellung sein. Die Fitnesslandschaftanalyse setzt sich mit dem problemunabhängigen Eigenschaftenraum auseinander (Reeves, 1999). Die Fitnesslandschaft L ist das Tupel bestehend aus der Menge aller Lösungen Ω eines Optimierungsproblems, der Fitnessfunktion Φ und einer Nachbarschaft N_k (Smith-Miles und Lopes, 2012). Φ weist jeder Lösung in Ω einen numerischen Wert zu. Dieser entspricht der Fitness einer Lösung. Die Nachbarschaft $N_k(u) = \{v \in \Omega : d(u,v) \leq k\}$ ist mithilfe der Distanzmetrik $d(u,v)$ definiert, die den Abstand zwischen den Lösungen u und $v \in \Omega$ angibt. Der Abstand d ist dabei, zum Beispiel, die Anzahl einer Anwendung eines elementaren Operators. d gibt also an, wie weit die Lösungen u und $v \in \Omega$ in Bezug auf einen elementaren Operator auseinander liegen. Die Fitnesslandschaft kann auch als Graph modelliert werden. Alle Lösungen Ω sind Knoten im Graph. Sind u und $v \in \Omega$ Nachbarn, existiert eine Kante im Graph zwischen u und v. Eine häufig verwendete Problemeigenschaft, basierend auf der Fitnesslandschaft, ist die Robustheit. Zur Berechnung der Robustheit wird die Autokorrelation der Lösungsqualitäten, aufgezeichnet durch einen Random Walk durch die Fitnesslandschaft, bestimmt. Eine weitere oft verwendete Eigenschaft für die Charakterisierung der Landschaft ist die *Fitness Distance Correlation* (FDC). Mithilfe der FDC wird versucht die Schwere eines Optimierungsproblems zu erfassen (Jones und Forrest, 1995). Weitere Eigenschaften für die Charakterisierung der Fitnesslandschaft sind, zum Beispiel, die Anzahl und die Verteilung der lokalen Minima, der Grad der Zufälligkeit der Lage des Optimums oder die Kolmogorov Komplexität als Maß für

die Strukturiertheit der Landschaft (Smith-Miles und Lopes, 2012). Mit der Fitnesslandschaft des TSP setzen sich intensiv, zum Beispiel, Stadler und Schnabl (1992) auseinander.

Zu beachten ist, dass die Fitnesslandschaft erst definiert ist, wenn die Menge der Lösungen Ω für die Problemstellung bekannt ist. Das bedeutet, dass Eigenschaften basierend auf der Fitnesslandschaft für die Analyse der Problemschwere oder für die Konstruktion von Algorithmen, nicht aber für die Algorithmenselektion geeignet sind (Smith-Miles und Lopes, 2012). Problemunabhängige Eigenschaften, die auch für die Selektion von Algorithmen herangezogen werden, basieren auf dem Konzept des Landmarking (Pfahringer et al., 2000). Hier werden effiziente und einfache Algorithmen auf die gegebene Problemstellung angewendet. Die Fitnesslandschaft konstruiert aus den schnell erzeugten Lösungen bildetet die Grundlage für die Generierung von Eigenschaften und für die Evaluation der Problemschwere.

Viele Veröffentlichungen im Bereich der Algorithmenselektion basieren ihre Studien auf problemabhängige Eigenschaften, auch oft bezeichnet als intrinsische Problemeigenschaften. Smith-Miles und Lopes (2012) evaluieren, neben weiteren Problemstellungen, auch Eigenschaften für das TSP und fassen Erkenntnisse über den Zusammenhang zwischen Eigenschaften und Problemschwere zusammen. Dabei zeigt sich, zum Beispiel, dass ein TSP, bei denen alle Distanzen zwischen den Knoten identisch sind, sehr leicht zu lösen ist, weil es $(n-1)!$ identische, optimale Lösungen gibt. Ein beliebiger Algorithmus findet also schnell optimale Lösungskandidaten. Eine hohe Varianz in der Kostenmatrix hingegen macht das Problem schwieriger zu lösen für exakte Lösungsalgorithmen (Cheeseman et al., 1991) und Heuristiken (Ridge und Kudenko, 2007). Es müssen also viele Lösungskandidaten evaluiert werden, bevor eine optimale Lösung gefunden werden kann.

Problemabhängige Eigenschaften für das TSP, die für Algorithmenselektion herangezogen werden, basieren überwiegend auf der Kostenmatrix C (vgl. Unterabschnitt 2.1.1). Mersmann et al. (2013) klassifiziert 47 bekannte Eigenschaften im Rahmen einer Untersuchung zur Performance des 2-Opt-Algorithmus in 8 Gruppen (vgl. Abschnitt 4.2). Diese Eigenschaften bilden die Grundlage für die in dieser Arbeit erarbeiteten Eigenschaften für das DVRP und werden abschließend im Folgenden kurz vorgestellt.

- *Distanz-Eigenschaften* bündeln alle problemabhängigen Eigenschaften, die von der Kostenverteilung der Knoten abhängig sind. Betrachtet werden, zum Beispiel, niedrigste, höchste oder durchschnittliche Kosten. Auch werden die Anteile der Knoten mit Kosten unter dem Durchschnitt und über dem Durchschnitt herangezogen. Zusätzlich werden Statistiken wie, zum Beispiel, die Standardabweichung und die Varianz der Kostenmatrix betrachtet. Eine weitere Eigenschaft, eingeordnet in die Kategorie *Distanz-Eigenschaften* ist die Länge einer zufälligen Tour. Für die Berechnung werden alle Kosten summiert und mit $2/(N-1)$ multipliziert.

- *Mode-Eigenschaften* gruppieren alle Eigenschaften, die den Modus der Kostenverteilung berücksichtigen. Zusätzlich werden verwandte Eigenschaften, wie die

Frequenz, die Anzahl und der Durchschnitt der Modi betrachtet.

- *Cluster-Eigenschaften* gruppieren alle problemabhängigen Eigenschaften, die auf den Ergebnissen von Clusteranalysen, mit unterschiedlicher Parametrisierung, der Knoten abgeleitet werden können. Typische Eigenschaften sind hier die Anzahl der Cluster und die durchschnittlichen Abstände der Knoten zu den Clustermittelpunkten.

- *Nächste-Nachbar-Distanz-Eigenschaften* bündeln Eigenschaften, die auf den Abständen der Knoten zum jeweils nächsten Nachbarn berechnet werden können. Auch hier werden das Minimum, das Maximum, der Durchschnitt und der Median betrachtet.

- Für Eigenschaften aus der Gruppe *Schwerpunkt-Eigenschaften* wird der geometrische Schwerpunkt der Probleminstanz berechnet. Statistiken wie das Minimum, das Maximum, der Durchschnitt und der Median, über die Abstände der Knoten zum berechnet Schwerpunkt bilden die Eigenschaften der Gruppe. Die Koordinate des Schwerpunkts ist ebenfalls eine Problemeigenschaft.

- *Minimaler Spannbaum (MST)-Eigenschaften* basieren auf dem berechneten minimalem Spannbaum. Betrachtet werden die Tiefe des Baums, aber auch das Minimum, das Maximum, der Durchschnitt und der Median der Kosten des Baums.

- *Winkel-Eigenschaften* gruppieren alle Eigenschaften, die auf der Grundlage der Winkel eines Knotens zu seinen beiden nächsten Nachbarn berechnet werden. Auch hier werden das Minimum, das Maximum, der Durchschnitt, der Median und die Standardabweichung betrachtet.

- *Konvexe Hülle-Eigenschaften* sind berechnete Statistiken über die konvexe Hülle der Probleminstanz. Betrachtet werden, zum Beispiel, die Fläche, die durch die Hülle aufgespannt wird, aber auch die Anzahl der Knoten, die Teil der Hülle sind.

2.2.3 Performanceraum

Der Performanceraum besteht aus der Menge der Performances Y, die sich aus den Lösungen von Probleminstanzen durch Algorithmen ergeben ($y(\alpha,x) \in Y$). Die Performance selbst kann dabei ein beliebiges Charakaristikum sein, mit dem sich Ergebnisse eines Algorithmus bewerten lassen. Typische Charakteristika sind, zum Beispiel, die Lösungsqualität, die Laufzeit des Algorithmus oder die Einfachheit der generierten Lösungen (Rice, 1976). Gemeinsam mit den Problemeigenschaften bildet die Performance die Grundlage für die Algorithmenselektion. Das Modell $S(f(x))$, welches auf der Grundlage von den Eigenschaften $f(x)$ einen Algorithmus α auswählt, basiert im Wesentlichen auf dem gelernten Zusammenhang zwischen Problemstellung und durch einen Algorithmus erzeugte Performance. Eine Menge von gelösten Probleminstanzen

bildet demzufolge die Grundlage für die Konstruktion des Modells. Dabei wirkt sich die Auswahl der Probleminstanzen für diese Trainingsphase des Modells wesentlich auf den Lernerfolg aus. Repräsentative Instanzen für einen Algorithmus sind modellabhängig. Im Kontext dieser Arbeit werden Probleminstanzen erarbeitet, bei denen die Performance eines Algorithmus minimal bzw. maximal ausgeprägt ist. Die Eigenschaften solcher Instanzen eigenen sich für das Training des in dieser Arbeit umgesetzten Modells, da so die wesentlichen Eigenschaften, die sich auf einen Algorithmus auswirken, bei der Trainingsphase berücksichtigt werden. Es zeigt sich, dass die zielgerichtete Generierung solcher Probleminstanzen geeigneter ist als zufällig generierte oder Benchmarkproblemstellung zu nutzen. Oft werden durch die gezielte Suche Probleminstanzen entdeckt, bei denen Algorithmen schlechtere oder bessere Performance aufweisen im Vergleich zu Benchmarkproblemstellung oder zufälligen Instanzen (van Hemert, 2006). Aus diesem Grund ist es ein üblicher Ansatz, mithilfe von, zum Beispiel, genetischen Algorithmen Probleminstanzen gezielt zu erzeugen und aufbauend auf diesen das Modell $S(f(x))$ zu konstruieren. Auch im Kontext der Analyse der Schwere eines Optimierungsproblems ist dieses Vorgehen verbreitet. Smith-Miles et al. (2010) und Mersmann et al. (2013), zum Beispiel, basieren ihre Studien auf zielgerichtet generierten Probleminstanzen. Gestaltet sich die Generierung von Instanzen aufgrund von komplexen Problemstellungen als schwierig, wird aber auch auf Benchmark- oder zufällig generierte Probleminstanzen zurückgegriffen.

Die zielgerichtete Generierung von Probleminstanzen kann unter den Begriff Data-Farming eingeordnet werden. Der Prozess des Data-Farmings bezeichnet das gesteuerte Erzeugen von Daten mithilfe von Experimenten oder Simulationen für die spätere Analyse (Sanchez, 2014). Die vorliegende Arbeit verfolgt ebenfalls den Ansatz Probleminstanzen zielgerichtet mithilfe eines Data-Farming-Experimentes zu generieren. Die Lösungen und die Eigenschaften der generierten Instanzen werden genutzt, um das Modell $S(f(x))$ zu implementieren. Sie bilden die Grundlage für die Identifikation von Problemeigenschaften, die sich auf die Performance der untersuchten Algorithmen auswirken.

2.2.4 Selektion von Algorithmen

Das Modell $S(f(x))$ bestimmt einen Algorithmus α, der für gegebene Eigenschaften $f(x)$ die Performance y maximiert und bildet die Grundlage für die Selektion von Algorithmen. Für die Realisierung des Modells gibt es verschiedene Ansätze. Einer der ersten, nach Wagner et al. (2018), erfolgreichen Implementierungen für SAT (vgl. Unterabschnitt 2.2.1) ist SATzilla, eingeführt von Xu et al., 2008. Das ASP Framework versucht ein empirisches Modell für jeden Algorithmus anzulernen, um mit diesem die Laufzeit von Algorithmen für Probleminstanzen voraussagen zu können. Basierend auf den Vorhersagen wird eine Entscheidung für einen Algorithmus getroffen. Bei der Online-Auswahl von Algorithmen, wird noch vor der Berechnung der Problemeigenschaften die Probleminstanz mit einer Menge von vorher definierten Algorithmen gelöst.

Jeder Algorithmus bekommt eine maximale Laufzeit zugewiesen. Kann eine Lösung schon mithilfe dieser Algorithmen gefunden werden, muss keine Algorithmenselektion mehr durchgeführt werden. So wird die Zeit für die Berechnung der Eigenschaften für besonders schnell lösbare Probleminstanzen eingespart. Maratea et al. (2014), zum Beispiel, wählen hingegen ein Klassifikationsmodell um den bestmöglichen Algorithmus für eine Probleminstanz zu selektieren. Das Modell lernt direkt die Zuordnung von Problemeigenschaften auf Algorithmus. Ein paarweises Klassifikationsmodell implementiert eine Erweiterung von SATzilla, eingeführt in Xu et al. (2011). Für jedes Paar von Algorithmen wird ein binärer Klassifikator trainiert. Es werden die Probleminstanzen für das Training herangezogen, bei denen sich die Performance der Algorithmen stark voneinander unterscheiden. Kadioglu et al. (2010) nutzen für die Implementierung des Modells $S(f(x))$ eine unüberwachte Clusteranalyse. Hier werden die Probleminstanzen in homogene Teilmengen unterteilt. Eine Probleminstanz, für die ein Algorithmus ausgewählt werden soll, wird einer der Teilmengen zugeordnet. Der Algorithmus, der durchschnittlich die besten Ergebnisse für die Probleme der Teilmenge generiert, wird ausgewählt (Wagner et al., 2018). Misir und Sebag (2013) interpretieren das Problem der Selektion eines Algorithmus als kollaboratives Filtern. Hier werden Verhaltensmuster von Gruppen ausgewertet, um auf Interessen einzelner Individuen zu schließen. Probleminstanzen werden auf der Grundlage von Eigenschaften in Gruppen eingeteilt. Eine Gruppe repräsentiert dabei die Vorliebe einer Instanz für einen bestimmten Algorithmus. Der Hauptunterschied zwischen kollaborativem Filtern und Algorithmenselektion ist, dass das Filtern auf der Extraktion von latenten Eigenschaften von Probleminstanzen beruht. Die typische Selektion von Algorithmen hingegen basiert auf beschreibenden Eigenschaften von Instanzen (Misir und Sebag, 2013). Die aktuell vielversprechendsten Ansätze sind, laut Wagner et al. (2018), das paarweise Klassifikationsmodell implementiert in SATzilla, eingeführt von Xu et al. (2011), und das AutoFolio System, eingeführt von Lindauer et al. (2015). AutoFolio kombiniert verschiedene Ansätze für die Modellbildung. Zum Beispiel, wird bestimmt, ob eine Cluster- oder eine Regressionsanalyse besser für die automatische Algorithmenselektion geeignet ist. Zusätzlich sucht AutoFolio automatisiert Parameter für das ausgewählte Modell.

Für die Algorithmenselektion in der vorliegenden Arbeit hat sich herausgestellt, dass die Realisierung des Modells $S(f(x))$ durch ein Regressionsmodell am besten geeignet ist. In Mayer et al. (2018*a*) zeigt der Autor dieser Arbeit, dass eine Regressionsanalyse einem Klassifikator für die zugrundeliegende Datenbasis überlegen ist. Das in dieser Arbeit konstruierte Modell basiert auf künstlichen neuronalen Netzen. Die Knoten (Neuronen) eines solchen Netzes sind in Schichten (engl. layer) angeordnet und oft in einer festen Hierarchie untereinander verbunden. Ausgehend von einer Eingangsschicht fließen die Informationen über eine oder mehrere Zwischenschichten (engl. hidden layers) bis zur Ausgangsschicht. Die Informationen auf der letzten Schicht stellen das Ergebnis dar. Bei den in dieser Arbeit betrachteten Feedforward-Netzen sind alle Knoten einer Schicht mit allen Knoten der Folgeschicht verbunden. Ein Neuron besteht im Wesentlichen aus einer Übertragungsfunktion (vgl. Gleichung 2.8) und einer Aktivierungsfunktion

(vgl. Gleichung 2.9). Die Übertragungsfunktion verarbeitet alle Informationen o_i aus den verbundenen Vorgängerneuronen. Dabei multipliziert die Funktion die Informationen mit den Kantengewichten w_{ij}, die die fließende Information abschwächen oder verstärken. Das Ergebnis der Übertragungsfunktion ist die Summe aller gewichteten Vorgängerinformationen und stellt die Eingabe net_j für die Aktivierungsfunktion dar. Die Aktivierungsfunktion verarbeitet das Ergebnis der Übertragungsfunktion net_j und gegebenenfalls einen Schwellwert θ_j und gibt das Ergebnis o_j an die Nachfolgeneuronen weiter. Die Ergebnisse der Aktivierungsfunktionen der Ausgabeschicht stellen das Ergebnis des künstlichen neuronalen Netzes dar.

$$net_j = \sum_{i=1}^{n} o_i w_{ij} \tag{2.8}$$

$$o_j = act_j(net_j - \theta_j) \tag{2.9}$$

In der Literatur werden verschiedene Aktivierungsfunktionen beschrieben. Eine häufig eingesetzte Funktion, die unter anderem auch bei der Modellbildung in dieser Arbeit verwendet wird, ist die RELU Funktion, eingeführt von Nair und Hinton (2010). Eine weitere Funktion, die in dieser Arbeit für die Neuronen der Ausgabeschicht zum Einsatz kommt, ist die Funktion SOFTMAX, eingeführt von Bridle (1990). Die SOFTMAX Aktivierungsfunktion erzeugt ein Ergebnis im Intervall von 0 bis 1 und wird üblicherweise in der Ausgabeschicht von neuronalen Netzen verwendet, die Multiklassenklassifikatoren oder Regressionsmodelle implementieren. Weiterführende Informationen über künstliche neuronale Netze und gängige Aktivierungsfunktionen sind, zum Beispiel, in Kriesel (2007) zu finden.

Das Trainieren eines neuronalen Netzes ist ein Optimierungsproblem. Während der Trainingsphase beim überwachten Lernen wird das Ergebnis durch die Berechnung eines Fehlers in Bezug auf das erwartete Ergebnis evaluiert. Das Optimierungsproblem besteht darin, den Fehler des erzeugten Ergebnisses durch das Anpassen der Kantengewichte w_{ij} und Schwellwerte θ_j zu minimieren. Das am häufigsten angewandte Optimierungsverfahren ist das Gradientenabstiegsverfahren, das im Backpropagation Algorithmus umgesetzt ist. In dieser Arbeit wird die Erweiterung „Adam“ des Gradientenabstiegsverfahrens, eingeführt von Kingma und Ba (2014), verwendet. Weiterführende Informationen über Backpropagation sind, zum Beispiel, in Hecht-Nielsen (1992) zu finden.

2.2.5 Anwendungsbeispiele

Untersuchungen hinsichtlich der Schwere von Problemstellung und Algorithmenselektion gibt es in vielen Disziplinen. Einen einführenden Überblick über Anwendungsbeispiele gibt bereits Unterabschnitt 2.2.1. Das folgende Unterkapitel beschränkt sich aus diesem Grund auf Anwendungen von Algorithmenselektion und Untersuchungen der Performance von Algorithmen im Kontext von dynamischen Fahrzeugwegeproblemen. Für eine umfangreiche Evaluation von Anwendungen in den verschiedenen Disziplinen sei auf Smith-Miles (2009) verweisen.

Frühe Arbeiten im Kontext von dynamischen dynamischen Fahrzeugwegeproblemen konzentrieren sich auf die Erarbeitung von intrinsischen Problemeigenschaften. Wie in Unterabschnitt 2.1.3 beschrieben, führen Lund et al. (1996) den DOD ein, um Probleminstanzen zu beschreiben. Larsen (2000) entwickelt auf Grundlage des DOD ein Framework für die Klassifikation von DVRP (vgl. Unterabschnitt 2.1.4). Im Zuge dieser Arbeit untersucht Larsen (2000) auch den Zusammenhang zwischen dem DOD und den Lösungskosten für das PDTRP. Eine Diskussion der beobachteten Korrelation zwischen Schwere der dynamischen Problemstellung und dem DOD im Kontext von Algorithmenselektion findet nicht statt. Eine evolutionäre Hyper-Heuristik, die Sequenzen von untergeordneten Heuristiken generiert, um das DVRP zu lösen, wird in Garrido und Riff (2010) eingeführt. Die Basis für die Sortierung der untergeordneten Heuristiken bildet die Performance der Heuristiken. Hier werden also problemunabhängige Eigenschaften für die Generierung der Reihenfolge der Operatoren herangezogen. Die vom Problem unabhängigen Eigenschaften werden von Garrido und Riff (2010) nicht im Kontext von Algorithmenselektion oder Problemschwere diskutiert. Eine weitere intrinsische Problemeigenschaft, die in Zusammenhang mit der Lösungsqualität steht, wird vom Autor dieser Arbeit in Mayer et al. (2017) erarbeitet. Die Auswahl von Algorithmen, basierend auf der eingeführten Eigenschaft, wird diskutiert, aber nicht umgesetzt. Eine erfolgreiche Algorithmenselektion, basierend auf der in Mayer et al. (2017) eingeführten Eigenschaft und weiteren, wird vom Autor dieser Arbeit erst in Mayer et al. (2018*b*) und Mayer et al. (2018*a*) beschrieben. Der Bezug zwischen den vorgestellten Veröffentlichungen und dieser Arbeit wird in den entsprechenden Kapiteln hergestellt.

2.3 Restriktionen und Definitionen

Der folgende Abschnitt diskutiert kurz die in dieser Arbeit betrachtete Variante des DVRP und fasst weitere definierte Restriktionen zusammen. Zusätzlich werden häufig verwendeten Begriffe mit Synonymen und Erläuterungen aufgelistet.

In der Literatur werden eine Vielzahl von komplexen Variationen des DVRP beschrieben (vgl. Unterabschnitt 2.1.4). Die vorliegende Arbeit konzentriert sich auf die Auswahl von Algorithmen auf der Grundlage von neu eingeführten Problemeigenschaften. Alle Eigenschaften werden mithilfe der Position der statischen und dynamischen Kunden berechnet und sind damit für alle Variationen des DVRP berechenbar (vgl. Abschnitt 4.1). Für die Erforschung der Problemeigenschaften und deren Beziehungen zu der Lösungsqualität von Algorithmen, konzentriert sich die vorliegende Arbeit auf die grundlegendste Variante des DVRP, das DTSP. Psaraftis et al. (2016) machen darauf aufmerksam, dass das DTSP in der aktuellen Literatur nur wenig Beachtung findet.

> *We have seen that the DVRP literature over the last few decades is full of approaches that have tackled ever more complex variants of DVRPs. Yet, to our knowledge, what seems to be the simplest variant of these problems remains unresolved. This is the DTSP, introduced in Psaraftis (1988).*
>
> - Psaraftis et al. (2016)

Das in dieser Arbeit betrachte DTSP berücksichtigt weder stochastische Anteile noch das Verschwinden von Kundenfragen. Es wird ausschließlich ein Depot und ein Fahrzeug mit unendlicher Kapazität modelliert. Zeiten für das Be- oder Entladen werden nicht berücksichtigt. Es werden keine Problemstellungen betrachtet, für die ausschließlich dynamische Kundenanfragen modelliert sind. Der DOD ist demzufolge für alle Problemstellungen im Intervall (0,1). Eine Anpassung existierender Routen kann ausschließlich bei geplanten Kunden erfolgen. Eine Änderung der Route zwischen Anfragen wird nicht betrachtet. Eine Lösung einer Probleminstanz minimiert die Gesamtkosten in Form von Fahrzeit, nach Garrido und Riff (2010) eine weit verbreitete Zielfunktion für DVRP. Die Berechnungszeit $t_{Berechnung}$ (vgl. Unterabschnitt 2.1.2) für die Ermittlung von Lösungen wird nicht berücksichtigt. Hier wird davon ausgegangen, dass Lösungen zur Verfügung stehen, noch bevor der nächste zu bedienende Kunde erreicht und bedient ist. Es zeigt sich, dass die Berechnungszeiten für die betrachteten Algorithmen, aufgrund von den hohen Rechengeschwindigkeiten der verwendeten Hardware, hinreichend klein sind.

Die folgende Tabelle 2.3 sammelt und erläutert Begriffe, die in dieser Arbeit häufig verwendet werden. Zusätzlich wird darauf eingegangen, welche Begriffe in dieser Arbeit synonym verwendet werden.

Tabelle 2.3: Zusammenfassung der in dieser Arbeit verwendeten Begriffe mit kurzen Erläuterungen und verwendeten Synonyme

Begriff	Erläuterung
Knoten	Mögliche ***Standorte*** / ***Orte*** im Problemraum mit zugeordneter Koordinate
Statischer Kunde	Oder auch ***statische Anfrage***, definiert eine ***Leistung***, die am zugehörigen ***Knoten*** erbracht werden muss; bildet die Planungsgrundlage für einen Planer
Dynamischer Kunde	Oder auch ***dynamische Anfrage***, definiert eine ***Leistung***, die am zugehörigen ***Knoten*** erbracht werden muss; werden erst während der Ausführung von geplanten ***Routen*** dem Planer bekannt
Kunden	Oder auch ***Anfragen***, sind die Gesamtheit aller ***dynamischen und statischen Kunden***
Leistung	Zu erbringende Dienstleistung: ***Lieferung***, ***Abholung*** oder ***Service*** bei einem ***Kunden***
Planungshorizont	Zeitraum, in dem Routen noch aufgrund von ***dynamische Kunden*** angepasst/umgeplant werden
Depot	Einem ***Knoten*** zugeordnet und Ausgangs- bzw. Endpunkt für eine ***Route***; üblicherweise Standort der Fahrzeugflotte und Lagerort für Ressourcen (benötigt für das Erbringen von ***Leistungen*** bei ***Kunden***)
Kante	Verbindung zwischen ***Knoten***; alle ***Knoten*** sind über ***Kanten*** mit allen anderen ***Knoten*** verbunden
Fahrzeug	Bewegen sich über ***Kanten*** und bieten Laderaum für Transporte (Abholungen, Lieferungen); können gegebenenfalls Services bei ***Kunden*** durchführen
Route, Rundreise	Menge von ***Kanten***; eindeutig zugeordnet zu einem ***Fahrzeug***; startet und endet bei einem ***Depot***; wird geplant, um ***Leistungen*** bei ***Kunden*** durchzuführen
Probleminstanz	Oder auch ***Problemstellung*** bezeichnet eine konkrete Ausprägung eines DVRP.
Eigenschaft	Oder auch ***Problemeigenschaft*** oder ***Metrik***; beschreiben eine Probleminstanz.

2.4 Zusammenfassung

Das Kapitel 2 legt die wissenschaftlichen Grundlagen für alle folgenden Kapitel. Intensiv wird in Abschnitt 2.1 das *Dynamic Vehicle Routing Problem* (DVRP) als Spezialfall des *Vehicle Routing Problem* (VRP) diskutiert. Es werden verschiedene Variationen des VRP mithilfe der aktuellen Taxonomie von Lahyani et al. (2015) eingeführt. Alle

diskutierten Ausprägungen der statischen Problemstellung können auch in einer dynamischen Umgebung angetroffen werden. In Unterabschnitt 2.1.3 werden existierende Methoden bzw. Metriken vorgestellt, die die Dynamik von Probleminstanzen beschreiben. Eine umfangreiche Diskussion zeigt, dass die Dynamik unzähliger Problemstellung durch die existierenden Problemeigenschaften nicht ausreichend beschrieben werden kann. Um Probleminstanzen voneinander zu unterscheiden, müssen neue Problemeigenschaften eingeführt werden. Die Dynamik hat einen wesentlichen Einfluss auf die Lösungsqualität von Algorithmen. Eine intensive Auseinandersetzung mit Problemgruppen in Unterabschnitt 2.1.4 und Lösungsansätzen in Unterabschnitt 2.1.5 zeigt, dass die Gruppierung der Problemstellungen keinen Bezug zu den anzuwendenden Algorithmen hat. Lösungsansätze werden für die unterschiedlichsten Probleminstanzen gruppenübergreifend erfolgreich angewandt. Dieses Ergebnis der Literaturrecherche unterstreicht die Notwendigkeit der Untersuchung von Beziehungen zwischen Problemeigenschaften und Lösungsqualitäten von Algorithmen.

Abschnitt 2.2 setzt sich intensiv mit dem *Algorithm Selection Problem* (ASP) auseinander. Der Rahmen für die vorliegende Arbeit, eingeführt in Abbildung 1.5, orientiert sich an einem Framework beschrieben für das ASP. Unterabschnitt 2.2.1 diskutiert Problemeigenschaften im Kontext des ASP und führt gleichzeitig Eigenschaften für das TSP ein, die für die Selektion von Algorithmen zum Einsatz kommen. Die hier vorgestellten Eigenschaften bilden die Grundlage für die in dieser Arbeit eingeführten Eigenschaften für das DVRP. Der Abschnitt 2.2 schließt mit der Diskussion von Modellen für die Selektion von Algorithmen in Unterabschnitt 2.2.4 und der Vorstellung von Anwendungsbeispielen in Unterabschnitt 2.2.5. Die an dieser Stelle vorgestellten Selektionsmodelle bilden die theoretische Grundlage für die im Zuge dieser Arbeit entwickelten Modelle.

Kapitel 3

Der quelloffene Rich Vehicle Routing Problem Simulator

Im nachfolgenden Kapitel wird der im Zuge dieser Arbeit entwickelte ereignisorientierte *Rich Vehicle Routing Problem Simulator* (RVRPSim) und das dazugehörige Simulationsmetamodell für *Rich VRP* (RVRP) vorgestellt (vgl. Unterabschnitt 2.1.1). Eine Vorgängerversion des Simulators wurde vom Autor dieser Arbeit in Mayer et al. (2016) eingeführt. Die Implementierung ist unter der Apache-Lizenz 2.0 auf Github veröffentlicht (Mayer, 2018). Für die nachvollziehbare Einordnung des Kapitels in den Gesamtzusammenhang der vorliegenden Arbeit wird in Abbildung 3.1 der realisierte Ansatz mit dem Fokus auf das Modell und die Simulation visualisiert. Die Simulation und das Modell bilden die Grundlage für das Erzeugen von Lösungen für Instanzen des DVRP.

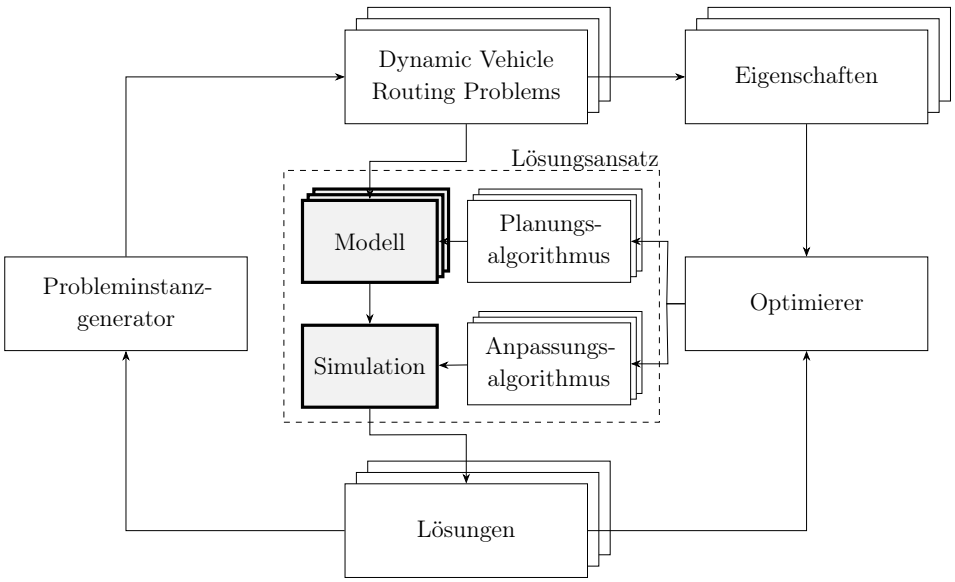

Abbildung 3.1: Lösungsansatz mit Fokus auf Modell und Simulation

Die Analyse der vorhandenen Modellierungsansätze zeigt, dass keine allgemein zugänglichen Repräsentationen für das DVRP existieren (vgl. Unterabschnitt 2.1.7).

Zusätzlich wird durch die Auswertung von aktuellen Arbeiten deutlich, dass nahezu in allen untersuchten wissenschaftlichen Veröffentlichungen, die sich mit dem Lösen von DVRP beschäftigen, eigene Simulationsumgebungen implementiert und genutzt werden. Meist sind diese Implementierungen wenig erläutert, sehr problemspezifisch konzipiert und stehen der wissenschaftlichen Gemeinschaft nur eingeschränkt zur Verfügung. Das Ergebnis der Analyse ist in Abbildung 3.2 visualisiert. Gut ist zu sehen, dass ausschließlich Gan et al. (2013) das untersuchte *Automated Guided Vehicles* (AGV) Problem mithilfe der kommerziellen Simulationssoftware Plant Simulation 9 (Plant 9) abbilden. Das AGV ist ein dynamisches PDP mit Zeitfenstern, bei dem Transportaufträge für autonome Fahrzeuge geplant werden müssen (vgl. Abschnitt 2.1.1). Die Autoren aller anderen untersuchten Arbeiten entwickeln dedizierte Simulatoren. Häufig werden diese kaum erläutert und nicht veröffentlicht. Es wird davon ausgegangen, dass Arbeiten, die keine spezifischen Angaben zum verwendeten Simulator machen, eigene Implementierungen für die Simulation nutzen.

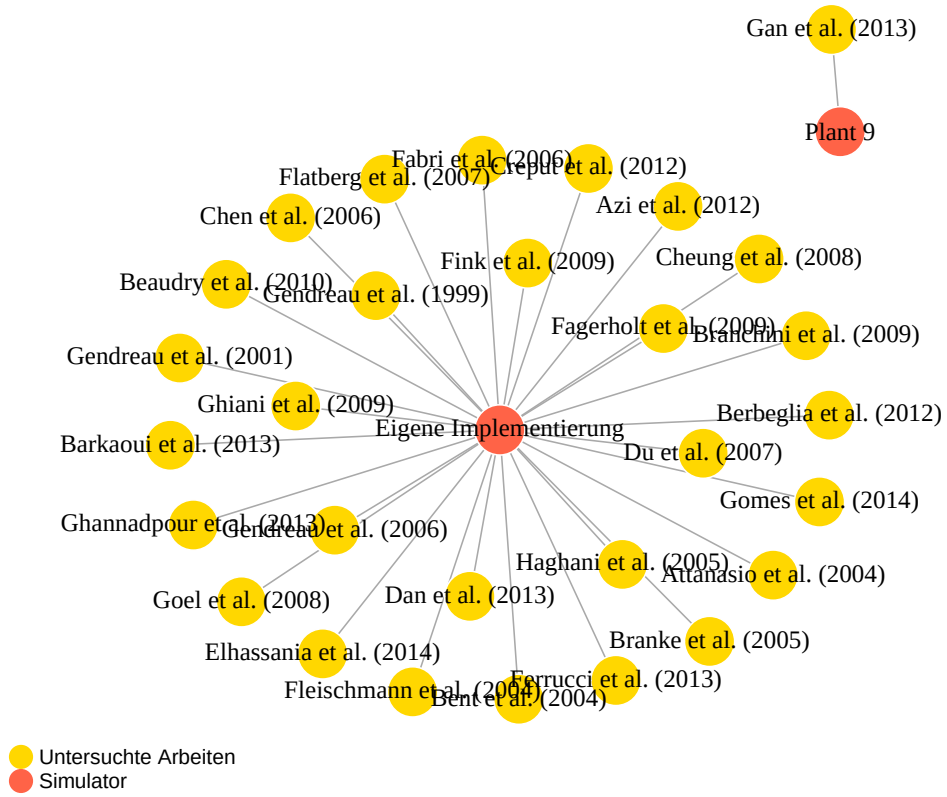

Abbildung 3.2: Zuordnung von verwendeten Simulatoren zu untersuchten Arbeiten

Um nicht noch eine weitere problemspezifische Repräsentation für das DVRP und einen nicht frei zugänglichen Simulator zu implementieren, wird im Zuge dieser Arbeit ein Simulationsmetamodell mit Simulator entwickelt, der als quelloffene Software der wissenschaftlichen Gemeinschaft frei zur Verfügung steht. Das Metamodell erlaubt die Realisierung verschiedenster Simulationsmodelle für die Simulation von unterschiedlichsten Varianten des DVRP. Für die Akzeptanz eines solchen Simulators muss das Metamodell eine Vielfalt unterschiedlicher Variationen des DVRP unterstützen bzw. geeignete Schnittstellen für Erweiterungen anbieten (vgl. Abschnitt 2.1). Zusätzlich ist es notwendig, einem potenziellen Nutzer des Simulators transparent darzulegen, welche Variationen des DVRP durch das Metamodell unterstützt werden, bzw. wo Erweiterungen zu implementieren sind, um zusätzliche Problemrestriktionen umzusetzen. Folgende Aufzählung fasst die Hauptanforderungen an den implementierten Simulator und das Metamodell zusammen.

- Unterstützung verschiedener Variationen des DVRP
- Leicht erweiterbar
- Transparente Aufbereitung von möglichen Erweiterungen

In Abschnitt 3.1 werden die von Law und Kelton (2000) identifizierten Grundkomponenten eines Simulators diskutiert und die entsprechenden Implementierungen im RVRPSim erläutert. Anschließend wird das Simulationsmetamodell im Detail vorgestellt. In Abschnitt 3.2 werden zusätzlich mithilfe des Metamodells umsetzbare Variationen des DVRP diskutiert. Das Metamodell wird aus diesem Grund nach der VRP-Taxonomy, eingeführt von Lahyani et al. (2015), klassifiziert (vgl. Unterabschnitt 2.1.1). Die Klassifikation soll einem möglichen Nutzer die durch das Simulationsmetamodell unterstützten Problemvariationen transparent darlegen. In Abschnitt 3.3 werden weitere implementierte Komponenten, die den Lösungsfindungsprozess unterstützen, wie, zum Beispiel, Modelltransformatoren oder Visualisierungen, kurz vorgestellt und das Kapitel zusammengefasst.

Für das gesamte Kapitel gilt, dass im Zuge dieser Arbeit in Quellcode umgesetzte Klassen oder Methoden mit der Schriftart Courier hervorgehoben sind. Zum Beispiel, wird die in Java umgesetzte Klasse mit dem Namen „Customer“ im Fließtext wie folgt formatiert: `Customer`.

3.1 Der Simulatorkern

Die meisten ereignisorientierten Simulationsmodelle teilen die wesentlichen Komponenten miteinander (Law und Kelton, 2000). Auch können die Interaktionen zwischen den Komponenten oft mithilfe der gleichen Logiken beschrieben werden. Grundsätzlich gilt für eine ereignisorientierten Simulation, dass Systemzustände durch Ereignisse zu bestimmten Simulationszeitpunkten verändert werden (Law und Kelton, 2000). Nach dem

im folgenden Unterabschnitt die Komponenten des Simulatorkerns erläutert wurden, wird in Unterabschnitt 3.1.2 auf die Logiken, die die Interaktionen der Komponenten beschreiben, eingegangen.

3.1.1 Komponenten

Im Folgenden werden die von Law und Kelton (2000) identifizierten Komponenten eines Simulationsmodells aufgeführt und kurz erläutert. Zusätzlich wird die Umsetzung der Komponenten im RVRPSim diskutiert.

Systemzustand

Der Systemzustand umfasst eine Sammlung von Variablen, die das System zu einem bestimmten Simulationszeitpunkt beschreiben. Angewendet auf das VRP können solche Variablen, zum Beispiel, aktuelle Orte von Fahrzeugen sein oder aber auch die Anzahl von Gütern in einem Lager. Im RVRPSim werden die den Zustand beschreibenden Variablen direkt an den Entitäten des Metamodells gespeichert. So wird, zum Beispiel, an der Entität Fahrzeug der aktuelle Ort und an der Entität Lager die aktuell vorhandene Anzahl von Gütern gespeichert. Auf das Simulationsmetamodell und die Entitäten wird in Abschnitt 3.2 genauer eingegangen.

Simulationsuhr

Nach Law und Kelton (2000) ist die Simulationsuhr eine Variable, deren Wert die aktuelle Simulationszeit widerspiegelt. Im RVRPSim wird die Simulationsuhr durch die Java Klasse `Clock` repräsentiert. Neben dem Schreiben und Lesen der Simulationszeit, bietet die Klasse zusätzlich die Möglichkeit die aktuelle Uhrzeit um eine Zeitspanne $\triangle t$ zu erhöhen.

Ereignisliste

Die Ereignisliste hält neben den Ereignissen auch die Simulationszeitpunkte, an denen diese auftreten. Das Abarbeiten eines Ereignisses eines bestimmten Typs verändert den Systemzustand. Im RVRPSim ist die Ereignisliste mithilfe der Wrapper-Klasse `Event List` implementiert. Eine Wrapper-Klasse hält als Attribut einen Java Datentypen und bietet zu diesem zusätzliche Funktionalität an. Die Klasse `EventList` hält eine Liste mit Ereignissen und bietet über die Serviceklasse `EventListService` zusätzliche Funktionalitäten für das Arbeiten mit der Liste an. Tabelle 3.1 fasst diese mit kurzen Erläuterungen zusammen.

Tabelle 3.1: In der Serviceklasse `EventListService` implementierte Funktionalitäten für das Arbeiten mit der Ereignisliste

Funktionalität	**Erläuterung**
`addEvent`	Fügt ein Ereignis der Ereignisliste hinzu.
`addEvents`	Fügt mehrere Ereignisse der Ereignisliste hinzu.
`getNextEvent`	Gibt das zeitlich nächste Ereignis zurück.
`getAndRemove NextEvent`	Gibt das zeitlich nächste Ereignis zurück und entfernt dieses aus der Ereignisliste.
`getNextOccurence ByType`	Gibt die Simulationszeit bis zum Eintreten des nächsten Ereignisses eines bestimmten Typs zurück.
`getRelativeTime TillOccurenceFor`	Bekommt die Simulationsuhr übergeben und berechnet die Simulationszeit bis zum Auftreten eines Ereignisses.

Statistische Zähler

Statistische Zähler sind nach Law und Kelton (2000) eine Menge von Variablen, die statistische Informationen über die Systemperformance speichern. Wie auch die den Zustand beschreibenden Variablen werden statistische Zähler direkt an den Entitäten des Modells gespeichert. So werden, zum Beispiel, an der Entität „Fahrzeug“ Informationen über zurückgelegte Strecken oder sonstige entstandene Kosten abgelegt. Auf das Simulationsmetamodell und die darin zur Verfügung stehenden Entitäten wird in Abschnitt 3.2 genauer eingegangen.

Initialisierungsroutine

Die Initialisierungsroutine initialisiert die Simulation zum Zeitpunkt Null. Im RVRPSim werden dazu alle initialen Ereignisse der Entitäten gesammelt und in die Ereignisliste überführt. Zusätzlich wird in der Routine das Modellverhalten durch den Planungsalgorithmus erzeugt (vgl. Abbildung 3.1 und Unterabschnitt 2.1.5).

Zeitroutine

Nach Law und Kelton (2000) ist die Zeitroutine ein Unterprogramm, dass das nächste abzuarbeitende Ereignis bestimmt und die Simulationsuhr entsprechend aktualisiert. Der RVRPSim setzt die Zeitroutine mithilfe der Java Methode `runStep` um. In dieser Methode wird die Funktionalität `getAndRemoveNextEvent` der Ereignisliste ausgeführt (vgl. Tabelle 3.1). Mit dem Entfernen des Ereignisses aus der Liste wird die Simulationszeit aktualisiert und der Systemzustand durch die Ausführung der Ereignisroutine angepasst.

Ereignisroutine

Die Ereignisroutine wird nach Law und Kelton (2000) ebenfalls in einem Unterprogramm realisiert. In der Routine werden Systemzustände bei der Abarbeitung von Ereignissen verändert. Für jeden Typ von Ereignis wird eine Ereignisroutine beschrieben. Im RVRPSim sind die Ereignisroutinen Parameter oder Bestandteil der Metamodellentitäten. Eine Entität gibt sich als Besitzer einer Ereignisroutine zu erkennen, in dem diese das Java Interface `IEventOwner` implementiert (vgl. Abbildung 3.4). Das Interface erzwingt die Implementierung der Methode `getInitialEvents`, über die die Startereignisse der Entität abgefragt werden (vgl. **Initialisierungsroutine**). Die Bearbeitung des Ereignisses wird mit der Methode `processEvent` angestoßen. Dabei werden die Systemzustände angepasst. Die Methode generiert als Rückgabeparameter möglich Folgeereignisse.

Bibliotheksroutine

Die Bibliotheksroutinen sind eine Menge von Unterprogrammen, die dazu genutzt werden, zufällige Beobachtungen aus Verteilungsfunktionen zu generieren und dem Simulationsmodell zur Verfügung zu stellen (Law und Kelton, 2000). Im RVRPSim werden alle Bibliotheksfunktionalitäten den Modellentitäten bei der Generierung zur Verfügung gestellt, unabhängig davon, ob diese davon Gebrauch machen.

Reportgenerator

Der Reportgenerator wird nach Law und Kelton (2000) in einem Unterprogramm realisiert und errechnet, nach Beendigung der Simulation, Statistiken auf der Grundlage der statistischen Zähler. Ein Reportgenerator ist nicht Bestandteil des RVRPSim. Vielmehr wird nach der erfolgreichen Simulation das gesamte Simulationsmodell zur Verfügung gestellt. Auf Basis der Systemzustände und statistischen Zähler, die Eigenschaften der Metamodellentitäten sind, können individuelle Reporte generiert werden.

Hauptprogramm

Das Hauptprogramm hat die Aufgabe die Zeitroutine aufzurufen, um das nächste abzuarbeitende Ereignis zu bestimmen. Anschließend wird nach Law und Kelton (2000) die Kontrolle an die entsprechende Ereignisroutine übergeben. Die Routine passt Systemzustände an und generiert mögliche Folgeereignisse. Das Hauptprogramm prüft das Abbruchkriterium der Simulation und stößt das Generieren des Reports an. Im RVRPSim wird das Hauptprogramm durch die Klasse `MainProgram` repräsentiert. Initialisiert wird `MainProgram` mit dem Simulationsmodell, einer Klassenstruktur bestehend aus Realisierungen der Metamodellentitäten, die in Abschnitt 3.2 im Detail vorgestellt werden. Die Initialisierungsroutinen der Modellentitäten werden bei diesem Vorgang angestoßen. Für die Steuerung der Simulation werden die Schnittstellen `run` und `runStep` angeboten. Die Methode `runStep` kann im Kontext einer Visualisierung

des Ablaufes der Simulation verwendet werden. Pro Ausführung von `runStep` wird ein Simulationsschritt ausgeführt, also genau ein Ereignis verarbeitet (delegiert an die Modellentitäten, vgl. **Ereignisroutine**). Eine Visualisierung kann so die Änderungen zwischen den Systemzuständen mithilfe der den Zustand beschreibenden Variablen an den Modellentitäten visualisieren. Mit der Schnittstelle `isSimulationFinished` kann durch die Visualisierung abgefragt werden, ob weitere Schritte zum Abarbeiten der Simulation notwendig sind. Die Steuerung der Simulation mit der Methode `run` ist für eine Visualisierung des Ablaufes der Simulation ungeeignet. Hier werden alle Ereignisse durch `MainProgram` abgearbeitet bis keine Ereignisse mehr in der Ereignisliste sind oder das Abbruchkriterium erreicht ist. Ein Abbruchkriterium kann, zum Beispiel, das Erreichen einer konkreten Simulationszeit sein. Die genauen Interaktionen zwischen den Komponenten des Simulatorkerns werden im folgenden Unterabschnitt beschrieben.

3.1.2 Interaktionen der Komponenten

Das in Abbildung 3.3 dargestellte Ablaufdiagramm, entnommen aus Law und Kelton (2000) und aus dem Englischen in das Deutsche übersetzt, visualisiert die Interaktionen zwischen den in Unterabschnitt 3.1.1 erläuterten Grundkomponenten. Der RVRPSim implementiert im Wesentlichen den in Abbildung 3.3 dargestellten Ablauf. Das Diagramm wird im Folgenden anhand der Umsetzung im RVRPSim vorgestellt.

Nach dem Start der Simulation, mithilfe der durch die Klasse `MainProgram` (vgl. **Hauptprogramm** in Unterabschnitt 3.1.1) zur Verfügung gestellten Funktionalitäten, werden alle Modellentitäten initialisiert. Die Initialisierung umfasst das Zurücksetzten der statistischen Variablen und des Systemzustands. Realisiert wird die Funktionalität mithilfe der Methode `reset`, implementiert am `IEventOwner` (vgl. Abbildung 3.4). Nach der Initialisierung werden die ersten Ereignisse durch die Modellentitäten generiert und in die Ereignisliste überführt. Die zyklische Abarbeitung der Ereignisse erfolgt nun in zwei Schritten. In einem ersten Schritt wird das als nächstes zu bearbeitende Ereignis bestimmt. Es definiert sich durch eine minimale Differenz zwischen der aktuellen Simulationszeit t_c und des Rückgabewertes der Methode `getTimeOfOccurence` implementiert am Ereignis (vgl. Abbildung 3.4). Existieren Ereignisse mit gleicher minimaler Differenz, wird das mit der höchsten Priorität zuerst abgearbeitet. Ist auch die Priorität gleich, wird das zuerst zu bearbeitende Ereignis zufällig ausgewählt. Nach der Auswahl wird die aktuelle Simulationszeit t_c auf die Simulationszeit t_o des ausgewählten Ereignisses gesetzt. In einem zweiten Schritt wird die Bearbeitung angestoßen. Die Bearbeitung erfolgt durch die Ereignisroutine `processEvent`, implementiert am Besitzer bzw. Erzeuger `IEventOwner` des Ereignisses. Auf die Routine wird im Zuge der Vorstellung des Simulationsmetamodells im folgenden Abschnitt 3.2 im Detail eingegangen.

Das Ergebnis der Ereignisroutine kann eine Liste neuer Ereignisse sein. Das `Main Program` setzt an allen Ergebnisereignissen die Simulationszeit t_o, an der das erzeugte Ereignis ausgelöst werden soll. Dabei wird die aktuelle Simulationszeit t_c und der

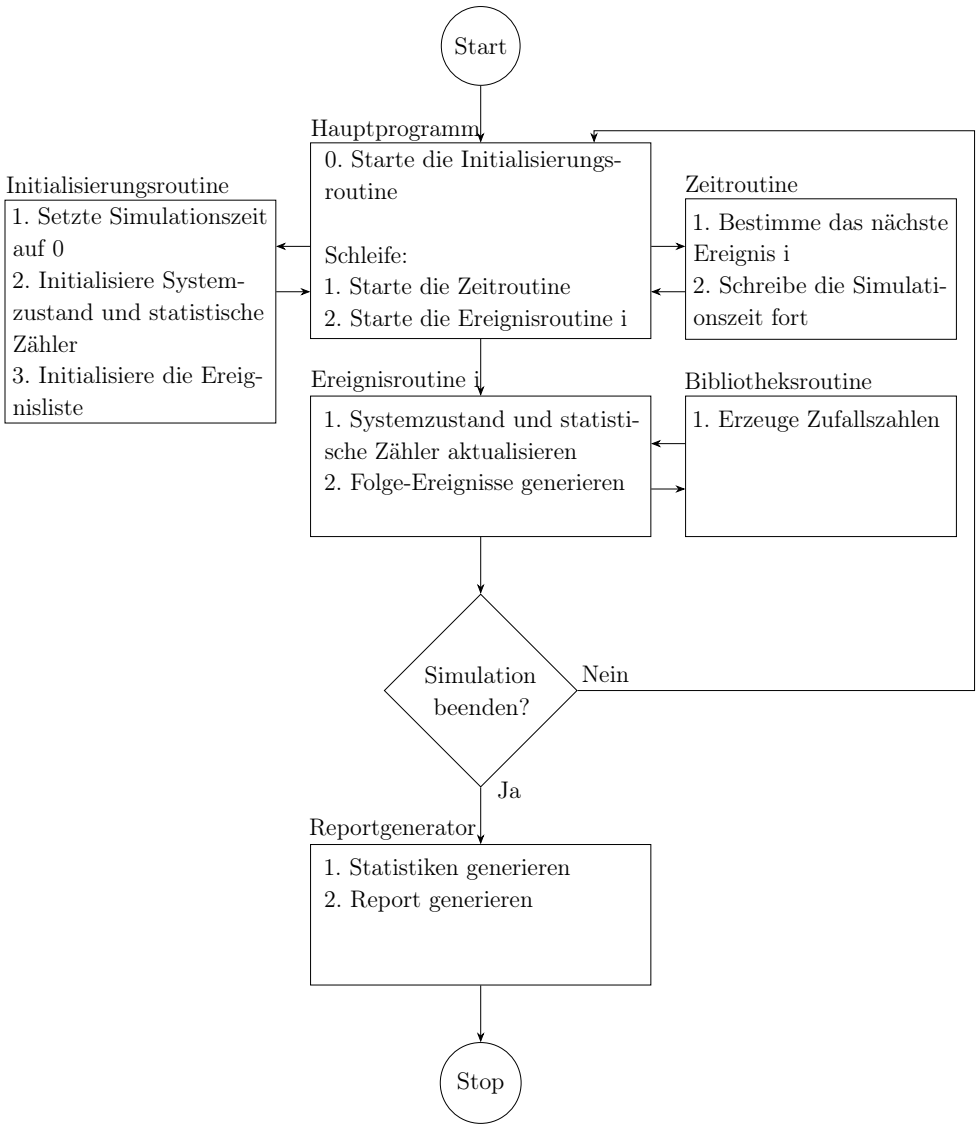

Abbildung 3.3: Ablaufdiagramm einer Diskreten Event Simulation (DES) angelehnt an Law und Kelton (2000)

Rückgabewert der Methode `getTimeTillOccurence` $\triangle t$ berücksichtigt. Die Simulationszeit t_o berechnet sich nach $t_o = t_c + \triangle t$. Wird nur ein einzelnes Ereignis durch die Routine als Ergebnis zurückgegeben, wird überprüft, ob es sich um ein Ereignis vom Typ `ERROR_EVENT` handelt. Das `ERROR_EVENT` signalisiert dem `MainProgram`, dass während der Bearbeitung ein Fehler aufgetreten ist und die Routine nicht vollständig abgearbeitet werden konnte. Befinden sich zum aktuellen Simulationszeitpunkt t_c weitere Ereignisse in der Ereignisliste, wird die Priorität des fehlerhaften Ereignisses erniedrigt und erneut in die Ereignisliste zum Zeitpunkt t_c eingefügt. Ist kein Ereignis zum aktuellen Simulationszeitpunkt t_c in der Liste vorhanden, wird die Simulation mit der Ausgabe eines Fehlers abgebrochen. Nachdem das Ergebnis der Routine verarbeitet ist, werden alle Modellentitäten, die sich ebenfalls für den entsprechenden Ereignistyp registriert haben, informiert. Entitäten, die an fremden Ereignissen interessiert sind, implementieren das Interface `IForeignEventListener`.

Abbildung 3.4 zeigt das im RVRPSim umgesetzte Ereignisinterfacemodell, das die Schnittstelle zwischen Simulatorkern und Simulationsmetamodell darstellt. Entitäten des Simulationsmetamodells implementieren die abgebildeten Interfaces, der Simulatorkern übernimmt die Steuerung durch die Aufrufe der entsprechenden Methoden.

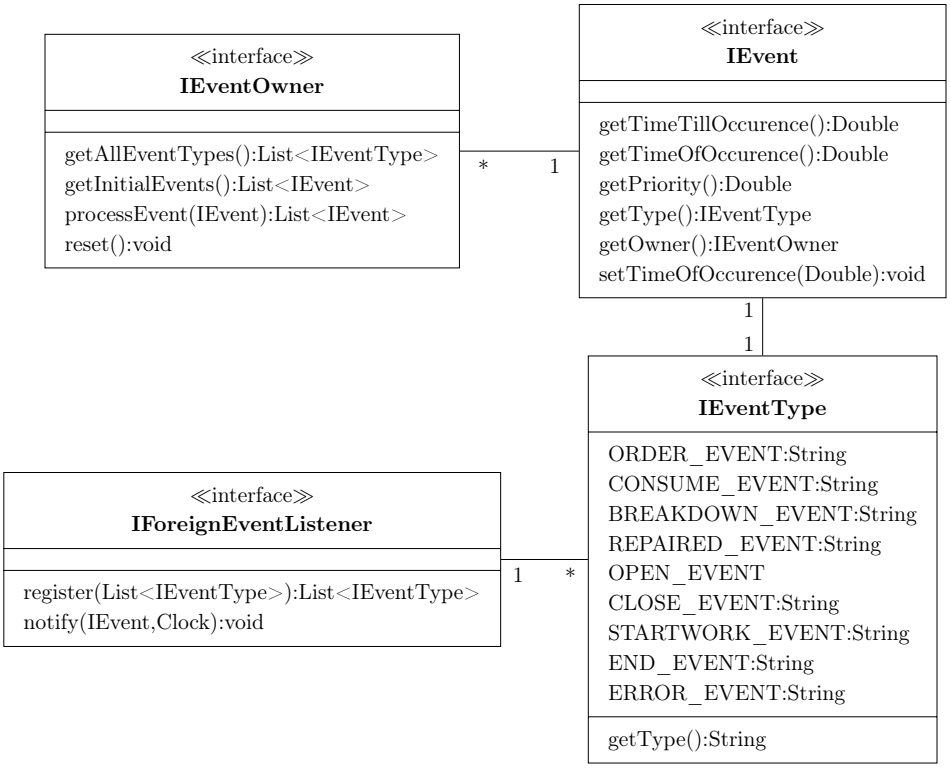

Abbildung 3.4: Klassendiagramm der im RVRPSim implementierten Ereignisinterfaces

Nachdem das bearbeitete Ereignis allen sich dafür interessierenden Entitäten bekannt gemacht worden ist, wird die Abbruchbedingung der Simulation überprüft. Die Simulation kann beendet werden, wenn eine vorher definierte Simulationszeit erreicht ist. Wird die Abbruchbedingung nicht erfüllt, startet die zyklische Abarbeitung der

Ereignisse aus der Ereignisliste erneut. Befindet sich kein Ereignis mehr in der Liste, wird die Simulation ebenfalls beendet. Nach Beendigung steht das Simulationsmodell mit verändertem Systemzustand und aktualisierten statistischen Zählern für eine Auswertung zur Verfügung. Der Reportgenerator, der eine Auswertung vornimmt, ist nicht Bestandteil des RVRPSim.

3.2 Das Simulationsmetamodell

Ein Simulationsmodell bildet die Struktur und das Verhalten eines realen Systems ab (Law und Kelton, 2000). Strukturelemente können Zustände einnehmen. Die Übergänge zwischen den Zuständen werden durch Verhalten gesteuert bzw. ausgelöst. Der RVRPSim stellt ein Simulationsmetamodell mit verschiedenen Entitäten für die Modellbildung zur Verfügung. Ein ausführbares Simulationsmodell, als Realisierung des Simulationsmetamodells, weist die in Abbildung 3.5 gezeigte Struktur auf. Das Modell (`VRPSimulationModel`) implementiert Repräsentationen von Verhalten (`BehaviourProvider`) und Struktur (`Structure`, `Network`). Zusätzlich bietet das Simulationsmodell Servicefunktionalitäten, die bestimmte Manipulationen des Modellverhaltens vereinfachen. Auf diese Funktionalitäten wird in Abschnitt 3.3 genauer eingegangen. Im folgenden Unterkapitel werden die implementierten Strukturelemente diskutiert, die das Simulationsmetamodell für eine Realisierung eines Simulationsmodells bereitstellt. Elemente des Netzwerkes, wie Straßen und Kreuzungen, sind Bestandteil der Strukturelemente, werden aber im RVRPSim gesondert in der Klasse `Network` zusammengefasst. Unterabschnitt 3.2.2 erläutert die Umsetzung des Verhaltens im Simulator. Grundsätzlich wird zwischen unabhängigem und abhängigem Verhalten unterschieden. Unabhängiges Verhalten verändert primär nur den Zustand eines Strukturelements und ist Bestandteil des Elements. Abhängiges Verhalten hingegen koordiniert Zustandsübergänge von mehreren Elementen der Struktur. Unterabschnitt 3.2.4 klassifiziert das implementierte Simulationsmetamodell nach der aktuellen Klassifikation von Lahyani et al. (2015).

3.2.1 Strukturelemente

Alle im RVRPSim umgesetzten Strukturelemente implementieren das Interface `IVRPSimulationModelElement` (vgl. Abbildung 3.6). Das Interface regelt die Interaktionen zwischen Strukturelementen, ausgelöst von Verhaltenselementen. So löst, zum Beispiel, das Verhalten Transport (vgl. Unterabschnitt 3.2.2) eine Interaktion zwischen Straße (`IWay`) und Fahrzeug (`IVehicle`) aus. Für die Durchführung des Verhaltens müssen beide Strukturelemente zur Verfügung stehen. Für die Dauer der Interaktion werden beide Elemente über das Verhaltenselement aneinander gebunden. So haben Strukturelemente die Möglichkeit unabhängig voneinander auf Interaktionen zu reagieren und die internen Zustände anzupassen. Eine Straße kann so, zum Beispiel, ab einer bestimmten Anzahl gleichzeitiger Interaktionen mit Fahrzeugen, die maximal Geschwindigkeit anpassen. Auf diese Weise können verlängerte Transportzeiten aufgrund von

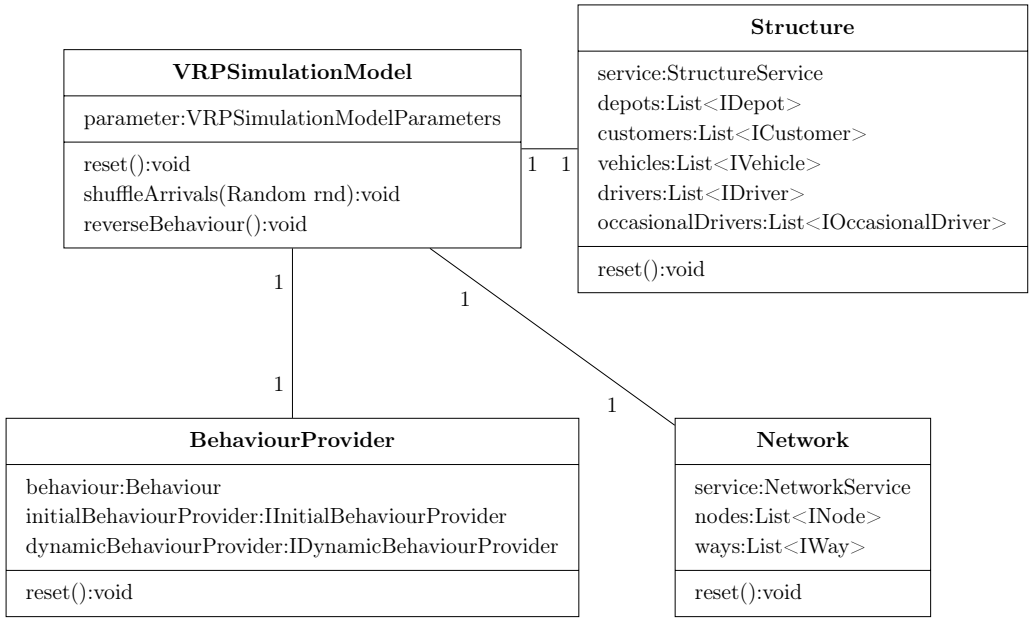

Abbildung 3.5: Klassendiagramm des im RVRPSim implementierten Simulationsmodells

Kapazitätsengpässen auf Straßen modelliert werden. Oder ein Kunde (`ICustomer`) kann, ab einer bestimmten Anzahl an Interaktionen mit Fahrzeugen, weitere Interaktionen verweigern. So können leicht Kunden mit begrenzter Kapazität für das Be- und Entladen modelliert werden.

Alle Netzwerkelemente implementieren die Interfaces `INode` oder `IWay` (vgl. Abbildung 3.6). Über die Knoten und Kanten kann ein Graph modelliert werden, auf dem sich Fahrzeuge, abhängig von Eigenschaften der Kanten und Knoten, bewegen können. Für die Abbildung eines vollständigen Graphen müssen im Simulationsmetamodell keine Kanten explizit modelliert werden. Das Routing der Fahrzeuge auf dem Graphen wird mit Verhaltenselementen modelliert, die in Unterabschnitt 3.2.2 vorgestellt werden. Die weiteren Strukturelemente, die nicht Teil des Netzwerkes sind, implementieren das Interface `IVRPSimulationModelStructureElement`. Über dieses Interface wird die Implementierung von Strukturelementparametern erzwungen. Das wichtigste Attribut der Parameter ist `home` vom Typ `IVRPSimulationModelNetworkElement`. Jedem Element der Struktur ist also ein Element des Netzwerkes zugeordnet. Über diese Zuordnung werden, zum Beispiel, die Standorte der Kunden oder Fahrzeuge modelliert. Die weitere Aufteilung der Strukturelemente erfolgt in Elemente, die sich auf dem Graphen bewegen können (*`Movable`) und Elemente, die einen Speicher bzw. Lager für zu transportierende Güter bereitstellen (*`WithStorage`). Ein Strukturelement kann beide Typen implementieren. Tabelle 3.2 zeigt die Strukturelemente und ordnet sie den unterschiedlichen Typen zu. Zusätzlich implementieren alle in Tabelle 3.2 aufgelisteten Elemente das Interface `IEventOwner` und weisen sich damit als Besitzer von eigenem Verhalten aus (vgl. Abschnitt 3.1 und Unterabschnitt 3.2.2).

Die in Abbildung 3.6 vorgestellte Interface-Struktur für Strukturelemente erlaubt die Modellierung einer Vielzahl von verschiedenen Variationen des DVRP. Der RVRPSim

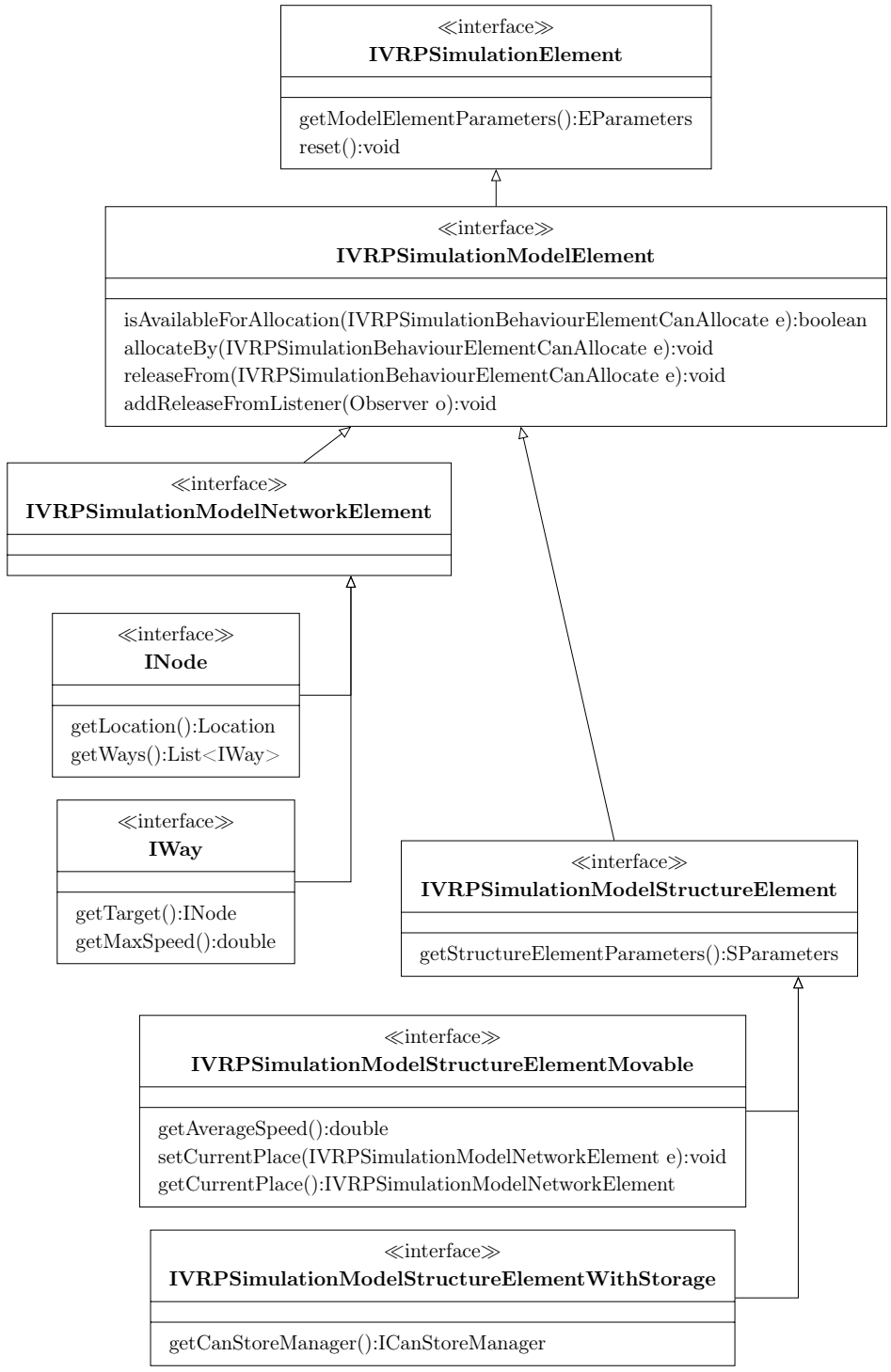

Abbildung 3.6: Klassendiagramm der im RVRPSim implementierten Interfacehierarchie der Strukturelemente

Tabelle 3.2: Implementierte Interfaces der Strukturelemente

Element	***Movable**	***WithStorage**	**IEventOwner**
IDriver	x	-	x
IVehicle	x	x	x
IOccasional Driver	x	x	x
ICustomer	-	x	x
IDepot	-	x	x

stellt eine Reihe von Standardimplementierungen als Entitäten des Simulationsmetamodells bereit, die im Folgenden kurz erläutert werden.

Node

Die Klasse `Node` implementiert das Interface `INode` und repräsentiert einen Knoten im Netzwerk. Der Knoten zeichnet sich im Wesentlichen durch eine dreidimensionale Koordinate, kodiert mithilfe der Klasse `Location`, aus und unterliegt sonst keinerlei Beschränkungen.

Way

Die Klasse `Way` implementiert das Interface `IWay` und repräsentiert eine Kante im Netzwerk. Eine maximale Geschwindigkeit kann über den Konstruktor der Klasse modelliert werden. Die Kante unterliegt sonst keinen weiteren Beschränkungen. So verändert sich, zum Beispiel, nicht die maximale Geschwindigkeit in Abhängigkeit von der Anzahl der sich auf der Kante befindenden Fahrzeuge. Eine solche Implementierung ist aber mithilfe der bereitgestellten Funktionalitäten im RVRPSim leicht zu realisieren.

Driver

Ein Fahrer wird im RVRPSim mithilfe der Klasse `Driver` repräsentiert. `Driver` implementiert die Interfaces `IDriver` und `IEventOwner`. Ein Fahrer hat keine eigenen Kapazitäten um Güter zu transportieren, kann aber auf dem Netzwerk bewegt werden. Der Fahrer zeichnet sich durch eigenes Verhalten aus. Das Verhalten des Fahrers ist im Interface `IDrivingHoursStrategy` gekapselt und modelliert Arbeitszeiten. Mithilfe von Ereignissen wird zwischen den Zuständen fahrbereit und nicht-fahrbereit gewechselt. So kann die Interaktionsbereitschaft mit anderen Strukturelementen, abhängig von der Simulationszeit, gesteuert werden. Die Bereitschaft zur Interaktion wird mithilfe der Methode `isAvailableForAllocation` kommuniziert.

SourceDepot

Die Klasse `SourceDepot` repräsentiert das Depot und implementiert die Interfaces `IDepot` und `IEventOwner`. Das `SourceDepot` kann eine unbegrenzte Anzahl von Gütern lagern und auch zur Verfügung stellen. Das Lager und die Steuerung des Lagers wird durch das Interface `ICanStoreManager` realisiert. Das Interface wird im Folgenden, nach der Auflistung der Standardimplementierungen, im Detail vorgestellt. Das Depot realisiert auch eigenes Verhalten, das im Interface `IOpeningHoursStrategy` gekapselt ist. Mithilfe des Verhaltens werden Öffnungszeiten modelliert. Über Ereignisse wird gesteuert, ob das Depot für Interaktionen mit anderen Strukturelementen zur Verfügung steht oder nicht, also gerade geöffnet ist oder nicht.

Vehicle

Ein Fahrzeug wird im RVRPSim von der Klasse `Vehicle` repräsentiert. Die Klasse implementiert das Interface `IVehicle`. Ein Fahrzeug kann auf dem Netzwerk bewegt werden und verfügt zusätzlich über begrenzte oder unbegrenzte Lagerkapazitäten für Güter. Das Lager und die Steuerung des Lagers wird durch das Interface `ICan StoreManager` realisiert. Das Interface wird im Folgenden, nach der Auflistung der Standardimplementierungen, im Detail vorgestellt. Eine maximale Geschwindigkeit, mit der das Fahrzeug auf dem Netzwerk bewegt werden kann, wird über den Konstruktor der Klasse konfiguriert. Auch ein Fahrzeug realisiert eigenes Verhalten, implementiert also auch das Interface `IEventOwner`. Die Logik des Verhaltens ist im Interface `IBreakdownStrategy` gekapselt. Ereignisse steuern die Übergänge zwischen den Fahrzeugzuständen fahrbereit und nicht-fahrbereit. Abhängig vom Zustand, steht das Fahrzeug für die Interaktion mit anderen Strukturelementen zur Verfügung.

Customer

Sowohl dynamische als auch statische Kunden werden im RVRPSim mithilfe der Klasse `Customer` abgebildet. Die Klasse implementiert die Interfaces `ICustomer` und `IEventOwner`. Ein Kunde hat eine feste Position auf dem Netzwerk und verfügt über begrenzte Lagerkapazitäten für Güter. Das Lager und die Steuerung des Lagers wird durch das Interface `ICanStoreManager` realisiert. Ein Kunde implementiert eine Reihe von eigenem Verhalten, die im folgenden Unterabschnitt 3.2.2 näher erläutert werden. Grundsätzlich verfügt der Kunde über Bestellverhalten, Verhalten, das die Erreichbarkeiten des Kunden abbildet und Verhalten, das den Konsum von gelagerten Gütern anstößt. Die Verhalten sind in den Interfaces `IOrderStrategy`, `IOpening HoursStrategy` und `IConsumeStrategy` gekapselt.

Das Transportieren und Lagern von Gütern stellt eine wesentliche Charakteristik von DVRP dar. Das Simulationsmetamodell stellt das Interface `ICanStore` für die

Realisierung von Lagern und das Interface `IStorable` für die Realisierung von Gütern zur Verfügung. Eine Entität kann sowohl ein Lager, als auch ein Gut selbst sein. So können, zum Beispiel, Güter in einer Palette, und die Palette in einem Fach gelagert werden. Die Palette ist in diesem Beispiel Lager und Gut. Der RVRPSim implementiert die genannten Güter und Lager mithilfe der Klassen `Good`, `Pallet` und `Compartment` (vgl. Abbildung 3.7). Ein Gut zeichnet sich durch einen Typ, einen Kapazitätsfaktor und einer Liste von Lagern aus, in der das Gut gelagert werden kann. Das Lager ist charakterisiert durch einen Typ, eine maximale Kapazität und eine Lagerpolitik, implementiert im Interface `ILoadingPolicy`.

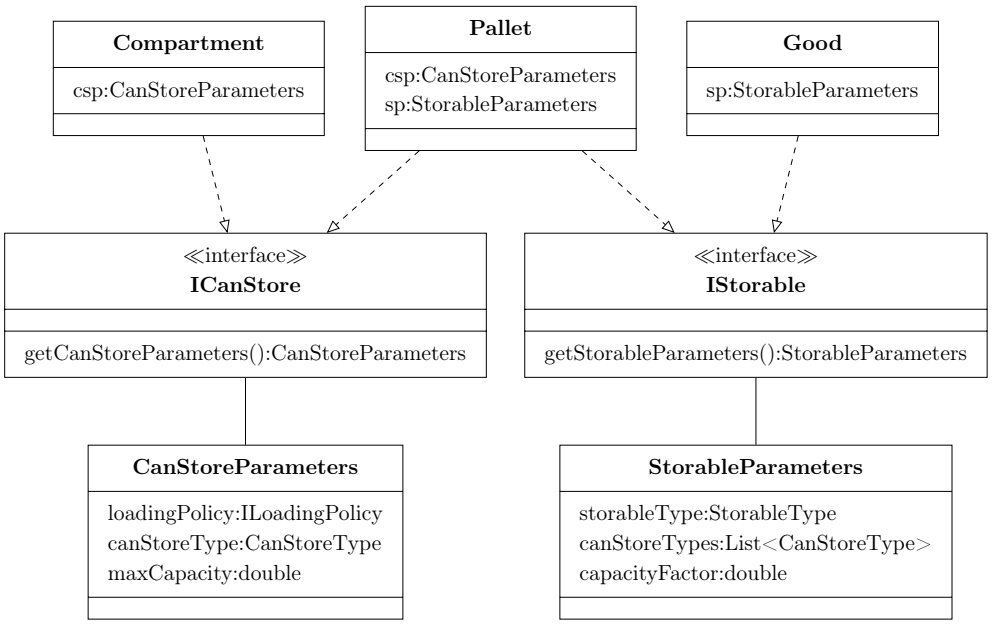

Abbildung 3.7: Klassendiagramm der Realisierung von Lagern und Gütern im RVRPSim

Eine Implementierung der Lagerpolitik übernimmt die Organisation vom Speichern und Freigeben von Gütern und bietet dazu eine Reihe von Funktionalitäten an (vgl. Abbildung 3.8). Im RVRPSim ist unter Anderem die Lagerpolitik *Last In First Out* (LIFO) implementiert (`LIFOLoadingPolicy`). Die als letztes eingelagerten Güter werden bei einer Anfrage zuerst ausgelagert. Zu beachten ist, dass die implementierte LIFO Lagerpolitik die Güter als Entitäten verwaltet. Ausschließlich eingelagerte Güter können dem Lager wieder entnommen werden. Anders verhält sich die Implementierung `Endless NoStateLoadingPolicy`. Hier können, unabhängig von eingelagerten Gütern, beliebige Mengen entnommen werden. Wenn das Lager leer ist, werden Güter durch einen Generator erzeugt.

Das Interface `ICanStoreManager`, dargestellt in Abbildung 3.9, stellt die Schnittstelle zwischen Strukturelementen mit Lagerkapazität (`*WithStorage`) und einem Lager mit dessen Gütern dar. Der Manager verwaltet ein oder mehrere Lager (`ICanStore`) und bietet Funktionalitäten für das Lagern und Auslagern. Zusätzlich können belegte oder noch freie Kapazitäten für Typen von Gütern abgefragt werden. Im Simulationsmetamodell ist unter anderem ein Manager implementiert, der ausschließlich eine `Endless`

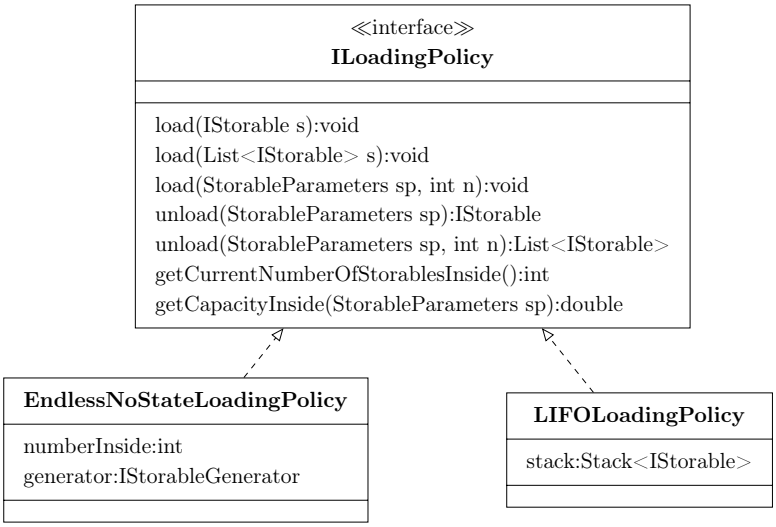

Abbildung 3.8: Klassendiagramm der im RVRPSim implementierten Lagerpolitiken

`NoStateLoadingPolicy` verwaltet (`SimpleNoStateCanStoreManager`). Hier muss keine Parametrisierung über den Konstruktor vorgenommen werden. Das verwaltete Lager kann beliebige Mengen von Gütern aufnehmen und bereitstellen. Die Implementierung `CanStoreManager` kann eine beliebige Anzahl von `LIFOLoading Policy` organisieren. Hier stehen nur bereits eingelagerte Güter für eine Auslagerung zur Verfügung. Das Strukturelement `SourceDepot` implementiert einen `Simple NoStateCanStoreManager`. Die Elemente `Vehicle` und `Customer` implementieren `CanStoreManager`.

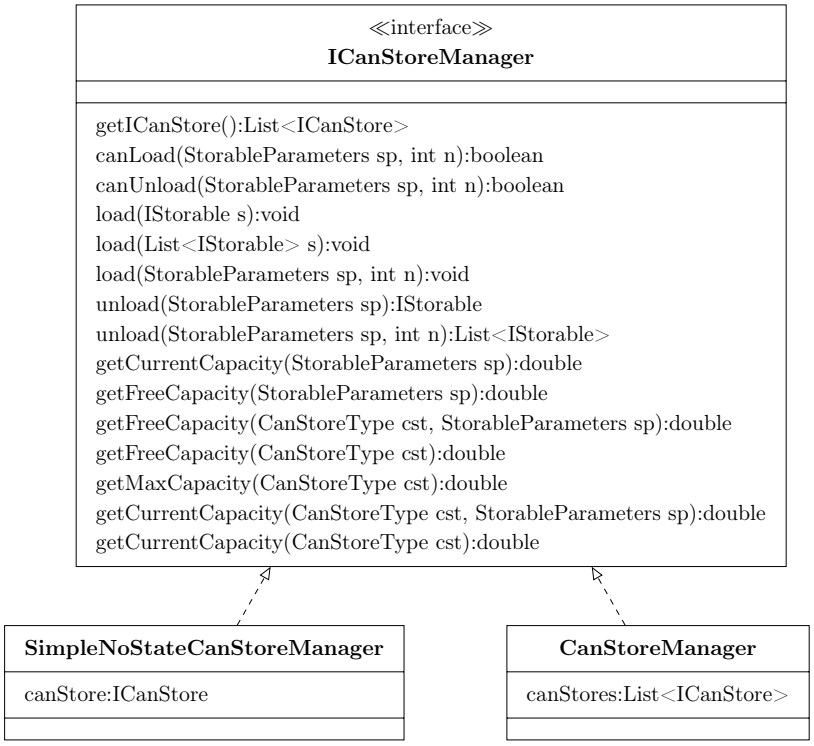

Abbildung 3.9: Klassendiagramm des im RVRPSim implementierten Lagermanagers

3.2.2 Verhaltenselemente

Ein Simulationsmodell wird sowohl durch Struktur als auch durch Verhalten repräsentierende Elemente charakterisiert (Law und Kelton, 2000). Dabei werden Zustände oder Eigenschaften der Strukturelemente durch Verhalten verändert oder angepasst. So ändert, zum Beispiel, das Verhalten „beladen“ den Lagerbestand des Strukturelements Fahrzeug. Im RVRPSim werden zwei verschiedene Arten von Verhalten unterschieden. Die erste Art ändert Eigenschaften oder Zustände von mehreren Strukturelementen und wird im Kontext des RVRPSim als **abhängiges Verhalten** bezeichnet. Das Verhalten „beladen“, zum Beispiel, ändert nicht nur den Lagerbestand des Strukturelements Fahrzeug, sondern zusätzlich auch den Lagerbestand des Elements Depot. Die zweite Art von Verhalten wird als **unabhängiges Verhalten** bezeichnet. Hier werden ausschließlich Eigenschaften bzw. Zustände von nur einem Strukturelement angepasst. Das Verhalten „schließen“ des Elements Depot, ist, zum Beispiel, ein Vertreter vom Verhalten der zweiten Art. Hier löst „schließen“ lediglich die Zustandsänderung vom Zustand „offen“ zu „geschlossen“ am Strukturelement selbst aus. Eigenschaften von weiteren Strukturelementen sind von **unabhängigem Verhalten** nur indirekt über **abhängiges Verhalten** betroffen. So wird sich, zum Beispiel, der Lagerbestand von Fahrzeug und Depot durch das Verhalten „beladen“ nicht verändern, wenn das Depot im Zustand „geschlossen“ ist. **Abhängiges Verhalten** ist also komplexes Verhalten, das von verschiedensten Eigenschaften und Zuständen von Strukturelementen abhängig ist und diese auch verändern kann. **Unabhängiges Verhalten** löst hingegen nur einfache

Zustandsänderungen von Strukturelementen aus.

Im Folgenden werden die im RVRPSim implementierten Verhaltenselemente getrennt nach den vorgestellten Arten erläutert.

Unabhängiges Verhalten

Die in Abbildung 3.10 dargestellten unabhängigen Verhalten kapseln die Logiken für die Erzeugung von semantisch zusammengehörigen Ereignissen, die Zustandsänderungen von Strukturelementen auslösen. So ist, zum Beispiel, das Interface `IOpeningHours Strategy` für das Erzeugen der Ereignisse verantwortlich, die das Öffnen und Schließen eines Strukturelements auslösen. Ein Hauptgrund für die Kapselung ist die im RVRPSim umgesetzte Trennung von Struktur- und Verhaltenselementen. Das Strukturelement kann verschiedene Zustände einnehmen. Die Änderungen der Zustände werden durch gekapseltes Verhalten ausgelöst. Die Organisation der Zustandsänderungen, also der Aufrufe der verschiedenen Verhaltenselemente, obliegt dabei den Strukturelementen. Das Element entscheidet abhängig vom aktuellen Zustand, welche Funktionalität von den Verhaltenselementen aufgerufen, also welche Zustandsänderung nach einer zeitlichen Verzögerung durchgeführt wird. Ist, zum Beispiel, ein Depot im Zustand „geschlossen“, so muss ein `OpenEvent` durch das gekapselte Verhalten erzeugt werden. Das erzeugte Ereignis definiert, wann die Zustandsänderung durchgeführt wird (vgl. Abbildung 3.4). Die Durchführung der Zustandsänderung liegt dann wiederum in der Verantwortung des Strukturelements. Mit der Kapselung des Verhaltens wird auch die Erzeugung aller zeitlichen Parameter gekapselt. Auf diese Weise können leicht identische Strukturelemente mit unterschiedlichem zeitlichen Verhalten modelliert werden. Die zeitlichen Parameter können deterministisch oder auch Realisierungen von Zufallsvariablen mit einer gegebenen Verteilungsfunktion sein.

Die Verhaltenselemente `IOpeningHoursStrategy`, `IDrivingHoursStrategy` und `IBreakdownStrategy` sind ähnlich aufgebaut. Hauptaufgabe der Elemente ist die zeitliche Steuerung der Übergänge zwischen zwei Zuständen mithilfe von Ereignissen. `IOpeningHoursStrategy` modelliert Öffnungszeit. Mithilfe des Elements `IDrivingHoursStrategy` können Arbeitszeiten abgebildet werden. Das Element `IBreakdownStrategy` realisiert Ausfallverhalten. Das Verhaltenselement `IConsume Strategy` modelliert den zyklischen Konsum von Gütern aus einem Lager. `IConsume Strategy` löst demzufolge keine Zustandsänderung, sondern eine Veränderung des Parameters Lagerbestand am Strukturelement aus. Sowohl die Anzahl der konsumierten Güter, als auch der Zeitpunkt des Konsums können deterministisch oder Realisierungen von Zufallsvariablen nach einer gegebenen Verteilungsfunktion sein. Auf diese Weise ist es möglich Schwankungen im Lager in Abhängigkeit von Bestellpolitiken und Güternachfragen zu modelliert. Das Element `IOrderStrategy` modelliert statische und dynamische Bestellungen eines Kunden (`Customer`). Eine Bestellung wird durch die Klasse `Order` repräsentiert. Statische Bestellungen werden unabhängig von Ereignissen bereitgestellt. Dynamische Bestellungen sind hingegen Bestandteil von Ereignissen.

Alle beschriebenen Verhaltenselemente werden im RVRPSim durch die Struktur-

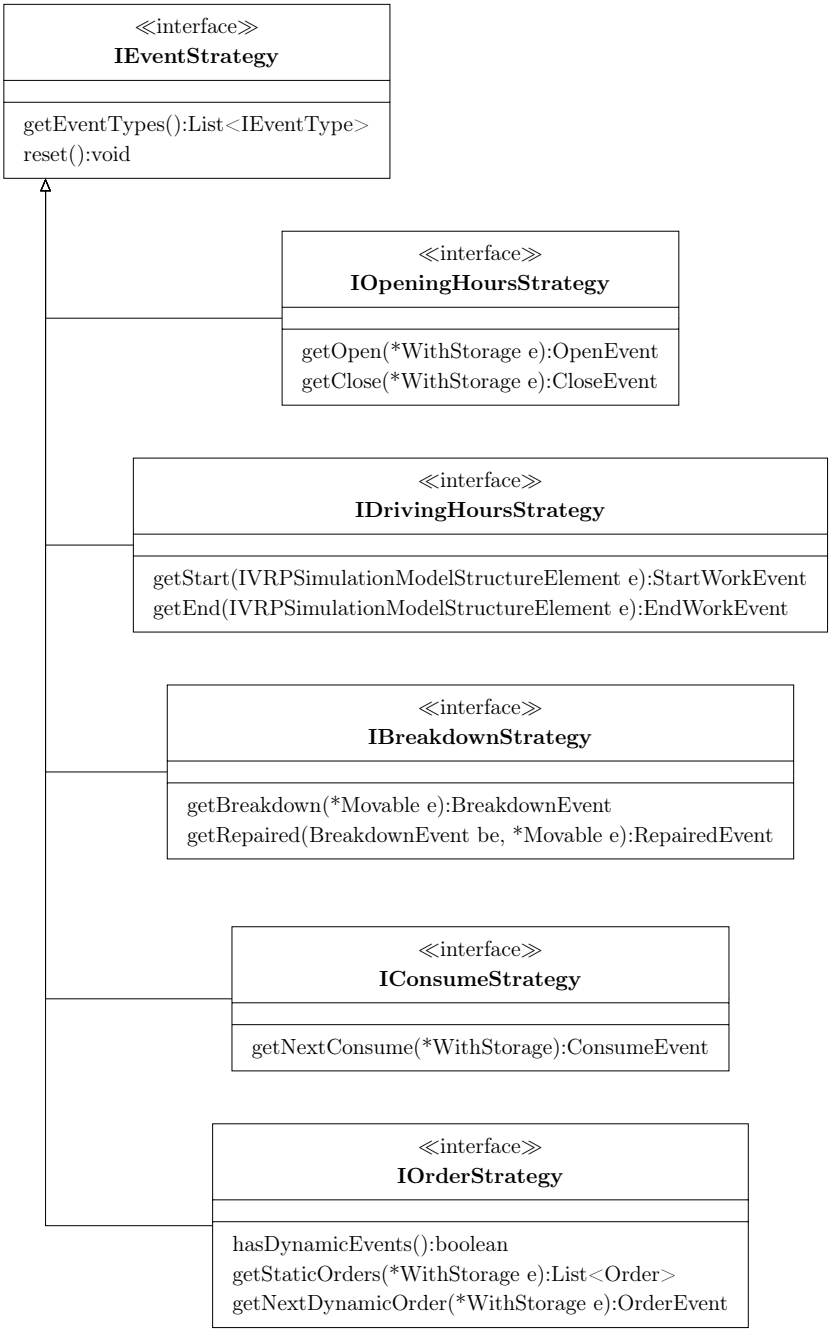

Abbildung 3.10: Klassendiagramm der im RVRPSim implementierten Elemente für unabhängiges Verhalten

elemente organisiert. Für eine Weiterentwicklung bietet sich an, die Organisation der Elemente durch dedizierte Manager zu realisieren, um den Grad der Trennung zwischen Struktur und Verhalten weiter zu erhöhen.

Abhängiges Verhalten

Neben dem unabhängigen Verhalten, das ausschließlich Zustände und Parameter von genau einem Strukturelement anpasst und Bestandteil des Strukturelements selbst ist, gibt es Verhalten, das von verschiedenen Eigenschaften und Zuständen von unterschiedlichen Strukturelementen abhängig ist und diese auch verändern kann. Im Kontext des RVRPSim wird dieses Verhalten als abhängiges Verhalten bezeichnet. Abhängiges Verhalten ist kein Bestandteil von Strukturelementen. Es ist in der Klasse `Behaviour` gekapselt. Diese ist eine Eigenschaft des `BehaviourProvider` (vgl. Abbildung 3.5). Die Klasse `Behaviour` repräsentiert die Gesamtheit des abhängigen Verhaltens in Form einer Menge von Realisierungen des Interfaces `ITour`. Das Interface repräsentiert eine Tour von einem Fahrzeug (`IVehicle`) mit einem Fahrer (`IDriver`) und organisiert die Interaktionen bzw. Verhalten zwischen Fahrzeug und Fahrer mit den verbleibenden Strukturelementen. Die Tour weist sich als Besitzer von Verhalten durch die Implementierung des Interfaces `IEventOwner` aus (vgl. Abbildung 3.11). Die Tour organisiert die Interaktionen oder auch Aktivitäten von Fahrer und Fahrzeug mit den verbleibenden Strukturelementen mithilfe des Interfaces `IActivity`. Die Eigenschaft `TourContext` speichert den aktuellen Zustand der Tour und eine Historie, über die auch Kosten der Tour abgeleitet werden können. Ausgehend von einer Startinteraktion werden durch die Tour alle Interaktionen von Fahrer und Fahrzeug mit den Strukturelementen sequenziell abgearbeitet. Aus diesem Grund besitzt jede Interaktion eine Folgeinteraktion (vgl. Abbildung 3.12). Die letzte Interaktion eines Fahrers und eines Fahrzeuges mit Strukturelementen hat keinen Nachfolger.

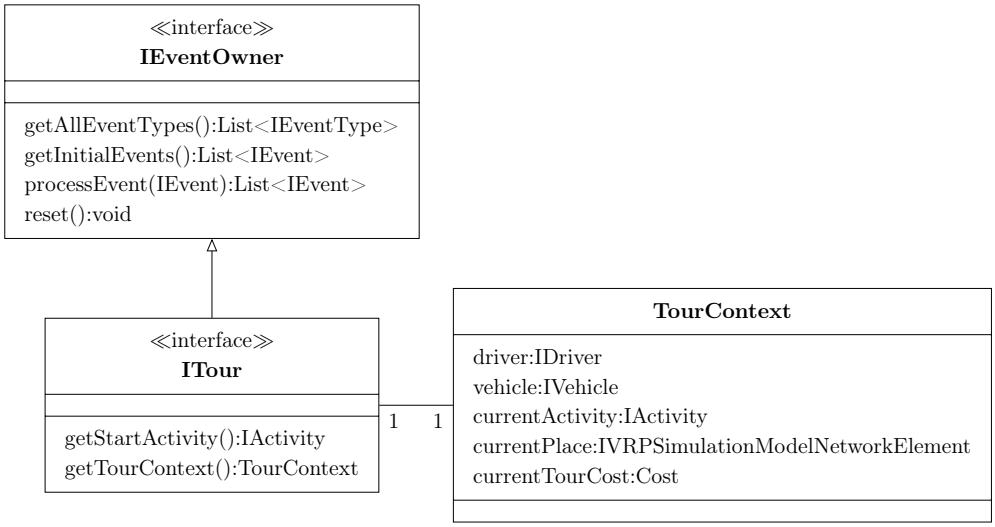

Abbildung 3.11: Klassendiagramm der Beziehungen der Verhaltenselemente `ITour` und `TourContext`

Im RVRPSim sind drei Interaktionen von Fahrzeug und Fahrer mit Strukturelementen implementiert (vgl. Abbildung 3.12). `LoadActivity` repräsentiert das Beladen und `UnloadActivity` das Entladen des Fahrzeuges. Beide Interaktionen werden durch den Typ und die Anzahl des zu be- oder entladenden Gutes charakterisiert. Zusätzlich muss an den Interaktionen der entsprechende Ladungspartner und eine Servicezeitrichtlinie definiert sein. Ein Partner kann dabei ein beliebiges Strukturelement mit Lager (`WithStorage`) sein. Mithilfe der Servicezeitrichtlinie `IServiceTimePolicy` wird die Dauer für das Be- und Entladen berechnet. Die Berechnung berücksichtigt den Fahrer, das Fahrzeug, den Ladungspartner und die Menge der zu be- oder entladenden Güter. Das `TransportActivity` repräsentiert das Fahren des Fahrzeuges und des Fahrers. Für den Transport muss sowohl ein Ziel (`IVRPSimulationModelNetwork Element`) als auch eine Routingrichtlinie (`IRoutingPolicy`) definiert werden. Eine Routingrichtlinie ermittelt den Weg zwischen aktuellem Standort und Ziel des Transports unter Berücksichtigung der aktuellen Simulationszeit. Zusätzlich wird die Zeit bestimmt, die für das Routing voraussichtlich verwendet wird. Im RVRPSim ist aktuell nur die `EuclideanDistanceIsTimeRouting` Routingrichtlinie implementiert. Diese Richtlinie sucht im Netzwerk nach dem kürzesten Weg zwischen Standort und Ziel unter Berücksichtigung der maximalen Geschwindigkeit des Fahrzeuges und der Strecke. Die Zeit, die für das Routing verwendet wird, ist proportional zu der Distanz des kürzesten Weges.

Alle `IActivity` implementieren eine Validierung. Dabei wird das Verhalten auf die Möglichkeit der Durchführung hin überprüft. Es wird, zum Beispiel, geprüft, ob für eine Beladeinteraktion der entsprechenden Ladungspartner auch am aktuellen Standort zur Verfügung steht. Sollte die Validierung nicht erfolgreich sein, liegt ein nicht durch die Simulation zu korrigierender, semantischer Modellierungsfehler vor und die Ausführung terminiert mit einer Fehlermeldung. Nach der Überprüfung kann die Interaktion mithilfe der Methode `prepareAction` vorbereitet werden. In der Vorbereitung wird die Dauer der Interaktion ermittelt. Für den Transport ist, zum Beispiel, die Dauer die Zeit die für den Transport aufgewendet wird. Zusätzlich werden während der Vorbereitung alle an der Interaktion beteiligten Strukturelemente dahingehend überprüft, ob diese für eine Interaktion zur Verfügung stehen. Realisiert wird das über die Nachfrage mithilfe der Methode `isAvailableForAllocation` an den Strukturelementen (vgl. Abbildung 3.6). Die Strukturelemente reagieren auf die Nachfrage in Abhängigkeit von ihrem internen Zustand. Stehen alle beteiligten Strukturelemente für die Interaktion zur Verfügung werden diese mithilfe der Methode `allocateBy` darüber informiert, dass eine Interaktion durchgeführt wird. Stehen nicht alle Elemente zur Verfügung, wird das betreffende Element mit dem Ergebnis der Vorbereitung `ActivityPrepareAction Result`, auch Bestandteil des Klassendiagramms in Abbildung 3.12, kommuniziert. Die Vorbereitung wird dann als nicht erfolgreich markiert. Nach der Vorbereitung erfolgt die Durchführung der Interaktion. Hauptbestandteil der Durchführung ist die Anpassung der Systemzustände und die Kommunikation über das Ende der Interaktion an die Strukturelemente mithilfe der Methode `releaseFrom` (vgl. Abbildung 3.6).

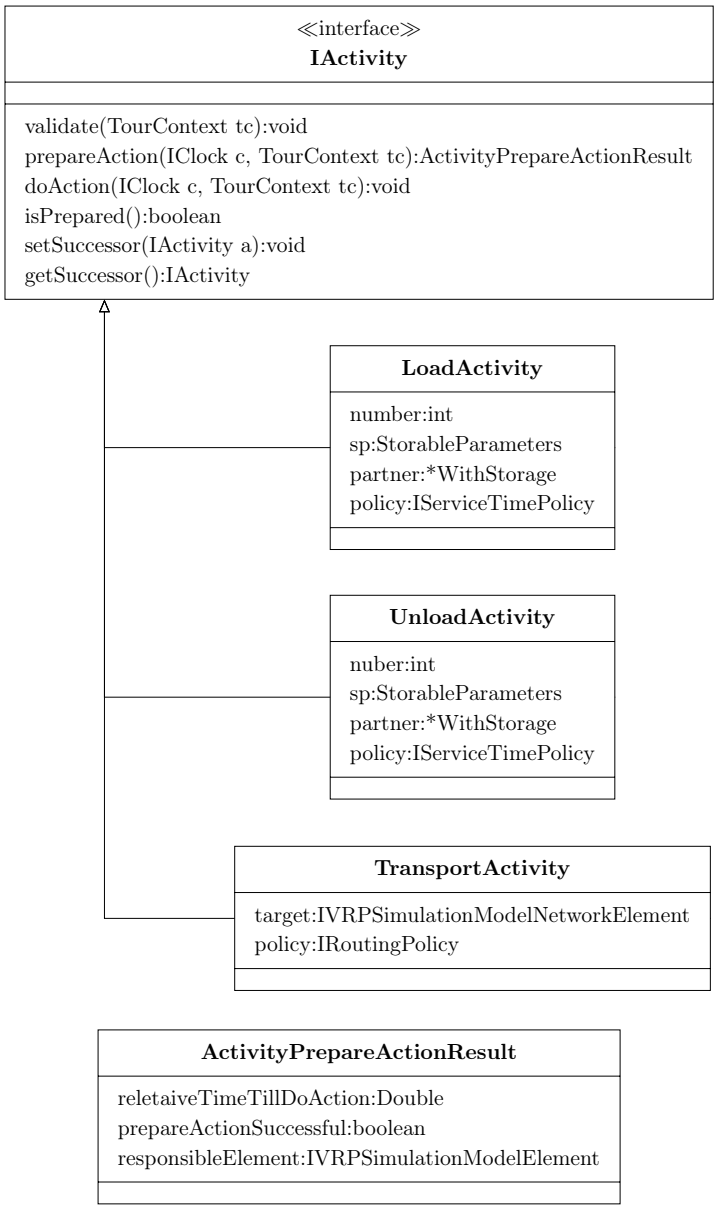

Abbildung 3.12: Klassendiagramm der im RVRPSim implementierten Elemente für abhängiges Verhalten

Die Tour ist unter anderem dafür verantwortlich sicherzustellen, dass zwischen Vorbereitung und Durchführung der Interaktionen die ermittelte Simulationszeit vergeht. Realisiert wird das von der Tour mithilfe von Ereignissen, die über das Interface `IEventOwner` mit dem Hauptprogramm kommuniziert werden (vgl. Abbildung 3.4). Ist die Vorbereitung einer Interaktion nicht erfolgreich (vgl. `prepare ActionSuccessful` in Abbildung 3.12), wird die Ausführung der Interaktionen blockiert. Die Tour ermittelt den möglichen Simulationszeitpunkt, an dem die Blockierung aufgehoben werden kann, und löst zu diesem Zeitpunkt eine erneute Vorbereitung aus. Die Differenz aus berechnetem Zeitpunkt und aktueller Simulationszeit wird als Wartezeit an den beteiligten Strukturelementen erfasst. Der Zeitpunkt kann mithilfe des Strukturelements, das für die Blockierung verantwortlich ist (vgl. `responsible Element` in Abbildung 3.12), und mit den von der Ereignisliste angebotenen Diensten (vgl. Tabelle 3.1) ermittelt werden. Das Ablaufdiagramm aus Abbildung 3.13 visualisiert die beschriebene Organisation der Interaktionen (`IActivity`) durch die Tour.

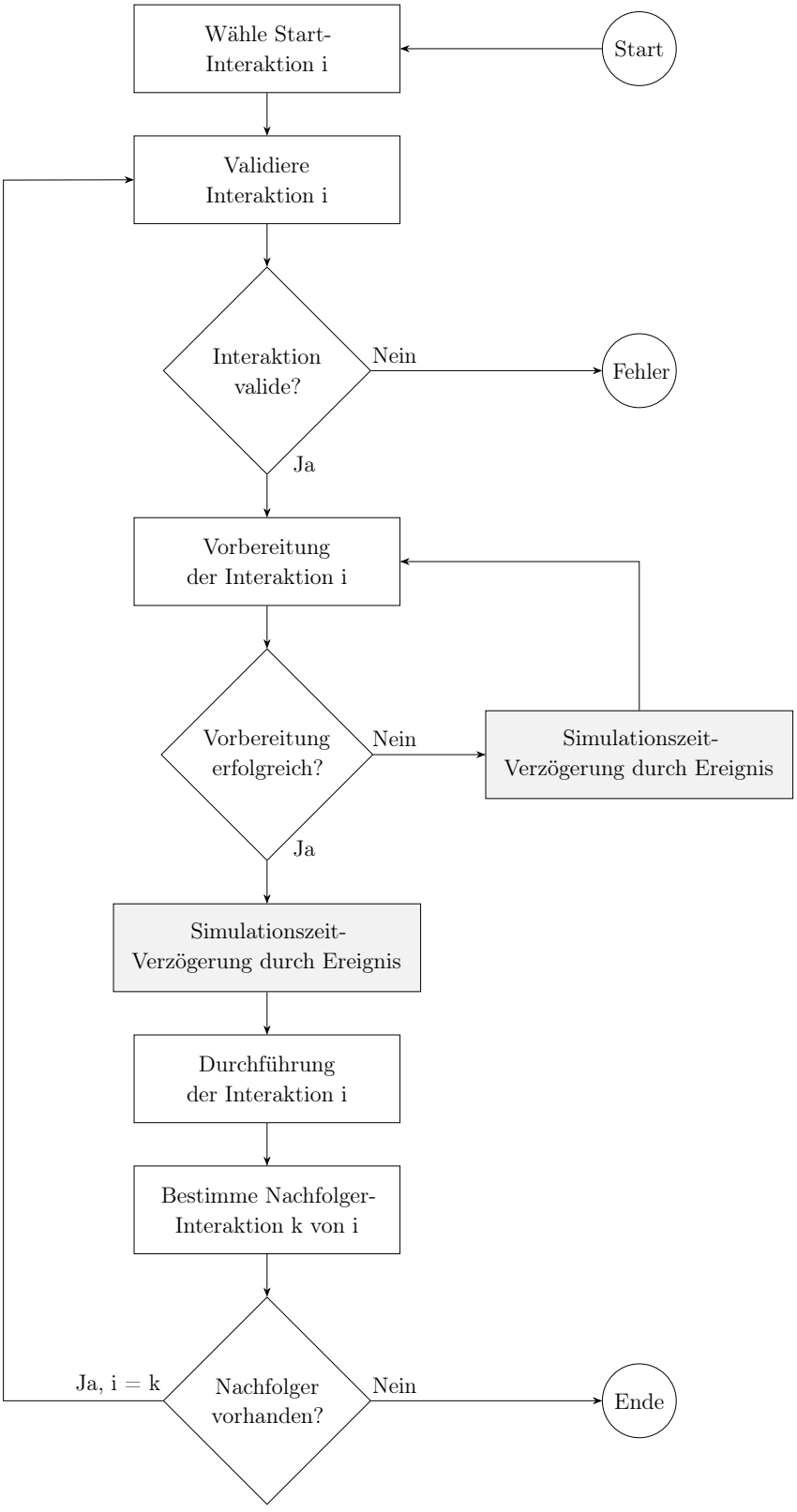

Abbildung 3.13: Ablaufdiagramm der Organisation der Interaktionen (`IActivity`) durch die Tour (grau hinterlegte Blöcke markieren Unterbrechungen bei der Ausführung)

3.2.3 Schnittstellen für Lösungsansätze

Wie in Unterabschnitt 2.1.5 beschrieben basieren Lösungsansätze für das DVRP überwiegend auf dem Prinzip der rollierenden Planung (Krypczyk, 2010). Dabei generiert ein Planungsalgorithmus eine Startlösung, die durch einen Anpassungsalgorithmus beim Auftreten von dynamischen Kundenanfragen angepasst wird. Im Kontext des RVRPSim ist eine Startlösung die Gesamtheit des abhängigen Verhaltens (vgl. Unterabschnitt 3.2.2). Ein Planungsalgorithmus muss also Sequenzen von Interaktionen für Touren generieren, die durch einen Anpassungsalgorithmus verändert werden. Abbildung 3.14 zeigt einen Auszug der Schnittstellen, die im RVRPSim für die Implementierung von Lösungsansätzen bereitgestellt werden.

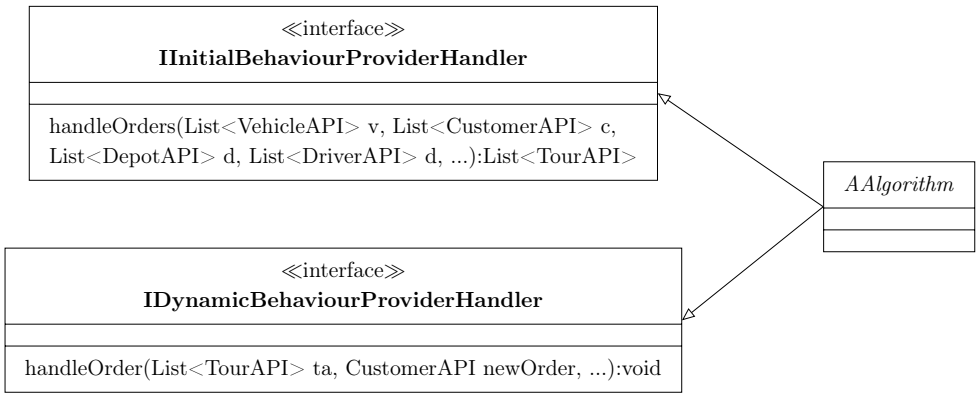

Abbildung 3.14: Klassendiagramm der im RVRPSim implementierten Interfaces für die Integration von Lösungsansätzen.

Die abstrakte Klasse `AAlgorithm` implementiert die Schnittstellen für einen Planungsalgorithmus (`IInitialBehaviourProviderHandler`) und einen Anpassungsalgorithmus (`IDynamicBehaviourProviderHandler`). Ein Planungsalgorithmus generiert Touren (`TourAPI`) für die bekannten Fahrer (`DriverAPI`) und Fahrzeuge (`VehicleAPI`). Die `TourAPI` ist durch eine sortierte Liste der Kunden (`CustomerAPI`) charakterisiert. Ein Planungsalgorithmus muss also lediglich, beim Nutzen der Schnittstelle `IInitialBehaviourProviderHandler`, für jeden Fahrer und Fahrzeug eine Teilmenge der Kunden bestimmen und diese in die Reihenfolge der gewünschten Abarbeitung bringen. Die Transformation der sortierten Liste von Kunden in abhängiges Verhalten (transportieren, be- und entladen) wird durch den RVRPSim übernommen. Ähnlich verhält es sich für eine Implementierung des Anpassungsalgorithmus. Diese muss ausschließlich den neuen dynamischen Kunden in eine bestehende Liste von Kunden einfügen. In der bestehenden Liste von Kunden, Bestandteil der `TourAPI`, befinden sich lediglich *„voll disponible“* Kunden, also Kunden, die im Laufe der Simulation noch nicht bedient worden sind (vgl. Unterabschnitt 2.1.5). Zu beachten ist, dass die dem Lösungsansatz übergebenen Struktur- und Verhaltenselemente nicht identisch mit den Strukturelementen des Simulationsmetamodells sind. `DriverAPI`, `VehicleAPI` und `CustomerAPI` sind lediglich Vereinfachungen der Strukturelemente, vorgestellt in

Unterabschnitt 3.2.1. Die Transformation der Elemente übernehmen Generatorklassen im RVRPSim.

Neben diesen vereinfachten Schnittstellen bietet der RVRPSim auch die Möglichkeit abhängiges Verhalten direkt zu erzeugen. Dazu müssen die Schnittstellen `IInitialBehaviourProvider` und `IDynamicBehaviourProvider` implementiert werden. Beide Interfaces sind Teil des Simulationsmetamodells (vgl. Abbildung 3.5) und implementieren Schnittstellen, die sich als solche zu erkennen geben. Zum Beispiel, implementiert der `IDynamicBehaviourProvider` das Interface `IForeignEventListener`, das gewährleistet, dass eine Implementierung über Ereignisse vom Typ Bestellung informiert wird. Auf eine genaue Erläuterung der komplexen Schnittstellen wird an dieser Stelle verzichtet. Eine Implementierung dieser ist nur dann sinnvoll, wenn das Simulationsmetamodell selbst erweitert wird und Erweiterungen bei der Generierung von abhängigem Verhalten relevant sind.

3.2.4 Klassifikation

Wie in der Einleitung in Kapitel 3 beschrieben, ist es für die Wiederverwendung des RVRPSim durch die wissenschaftliche Gemeinschaft wichtig, transparent aufzuarbeiten, welche Variationen des DVRP mit dem Simulator untersucht und wie Erweiterungen integriert werden können. Aus diesem Grund werden im Folgenden Umsetzungsmöglichkeiten von verschiedenen Variationen des DVRP mithilfe des RVRPSim diskutiert. Die VRP-Taxonomie von Lahyani et al. (2015) dient der Strukturierung der Diskussion und der Klassifikation des Simulators. Für eine umfangreiche Erläuterung der Taxonomie wird an dieser Stelle auf Unterabschnitt 2.1.1 verwiesen. Im Anschluss an die Diskussion wird diese in der Tabelle 3.3 kurz zusammengefasst und der Umsetzungsgrad der Taxonomie im Simulator aufgezeigt.

Eingabedaten

Der RVRPSim unterstützt die Abbildung von Eingabedaten mit Unsicherheiten. Sowohl schwankende Kundenbedarfe, Öffnungs-, als auch Transportzeiten können leicht umgesetzt werden. Für die Abbildung von Unsicherheiten in Eingabedaten müssen Strategien implementiert werden. Diese müssen die bei der Erzeugung von Ereignissen benötigten Werte, zum Beispiel, Zeiten oder Mengen, als Realisierungen von Zufallsvariablen nach einer gegebenen Verteilungsfunktion erzeugen (vgl. Unterabschnitt 3.2.2 und Abbildung 3.10). Aktuell sind im RVRPSim ausschließlich deterministische Strategien implementiert.

Für die Abbildung von schwankenden Transportzeiten ist es notwendig eine Routingrichtline zu implementieren, die bei der Berechnung der Transportzeit stochastische Schwankungen berücksichtigt. Eine neue Routingrichtlinie wird ohne Erweiterungen des RVRPSim durch das `TransportActivity` verarbeitet. Eine Strecke zwischen Standort und Ziel eines Transporters kann unter Berücksichtigung der Simulationszeit berechnet werden. So können schwankende Transportzeiten in Abhängigkeit von Tages-

zeiten modelliert werden. Schwankende Servicezeiten beim Be- und Entladen können ähnlich mithilfe einer neuen Implementierung von `IServiceTimePolicy` realisiert werden.

Ein weiterer Aspekt der Eingabedaten ist die Evolution der Probleminformationen. Hier unterstützt der RVRPSim das Erscheinen neuer Kundenanfragen während der Simulation mithilfe dynamisch erzeugter Bestellereignisse. Weitere, sich über die Zeit ändernde Probleminformation, werden durch den Simulator nicht berücksichtigt.

Entscheidungsmanagement

Der RVRPSim unterstützt verschiedene Variationen des klassischen VRP. Es können aber auch Strategien basierend auf den Lagerbeständen von Kunden erprobt werden, da mithilfe einer `IConsumeStrategy` auch das Verbrauchsverhalten von Kunden abgebildet werden kann (vgl. Abbildung 3.10). Entscheidungen über das Routing der Fahrzeuge hinaus werden durch den Simulator nicht direkt unterstützt. Grundsätzlich können aber Routingstrategien bei unterschiedlichen Platzierungen von Flotte oder Depot in verschiedenen Simulationsläufen mit unterschiedlichen Modellen untersucht werden.

Anzahl der Depots

Mithilfe des RVRPSim kann eine beliebige Anzahl von Depots abgebildet und simuliert werden.

Operationstyp

Sowohl Lieferungen als auch Abholungen können mithilfe einer `Order`, ausgelöst vom Kunden, abgebildet werden. Eine `Order` mit der Menge Null, kann zusätzlich dazu dienen, eine Forderung nach einem Service zu modellieren. Bestellungen, die den Transport eines Gutes oder einer Person von einem Ort zu einem konkreten Ziel abbilden, werden vom RVRPSim ebenfalls unterstützt. Mithilfe von Parametern am Element `Order` kann das Ziel einer Bestellung definiert werden. Die Analyse von *Pick-up and Delivery Problem* (PDP) kann so auch mithilfe des implementierten Simulators realisiert werden.

Ladungsteilung Restriktionen

Der RVRPSim sieht keine Begrenzungen der Fahrzeuge bei der Bedienung von Bestellungen durch Kunden vor. So können verschiedene Fahrzeuge den Bedarf einer Bestellung befriedigen.

Planungsperiode

Die Simulation terminiert, wenn keine Ereignisse mehr abzuarbeiten sind oder eine vorgegebene Simulationszeit erreicht ist (vgl. Abschnitt 3.1). So können beliebig viele

Perioden simuliert und auch Langzeit-Fragestellungen mithilfe des RVRPSim untersucht werden.

Mehrfachnutzung der Fahrzeuge

Fahrzeuge können für unterschiedliche Touren geplant und damit mehrfach genutzt werden. Interne, die Validität der Touren sichernde Mechanismen verhindern dabei, dass Fahrzeuge gleichzeitig in Touren aktiv sind.

Fahrzeuge

Die Klasse `Vehicle` kann durch Parametrisierung Fahrzeuge mit verschiedenen Lagerkapazitäten, Geschwindigkeiten oder möglichen Arten von zu transportierenden Gütern abbilden. Das Simulationsmetamodell beschränkt sich also nicht auf die Abbildung einer homogenen Transportflotte. Auch sind Ausfälle von Fahrzeugen im Simulator vorgesehen. Durch die Kapselung von Fahrzeug und Lager können leicht Fahrzeuge mit verschiedenen Transportbereichen realisiert werden. Zusätzlich sind Lagerpolitiken im RVRPSim vorgesehen (vgl. Abbildung 3.8).

Fahrzeuge und Anhänger werden im Simulator nicht getrennt voneinander betrachtet. So muss für die Abbildung eines *Truck and Trailer Routing Problem* (TTRP) das Simulationsmetamodell angepasst werden. Dazu ist es sinnvoll die Tour dahingehend zu erweitern, dass diese nicht nur genau ein Fahrzeug und einen Fahrer, sondern eine Liste von Fahrzeugen mit Fahrer verwaltet. Zusätzlich müssen weitere Aktivitäten eingeführt werden, die das Ab- und Ankoppeln der Fahrzeuge modellieren und den Systemzustand entsprechend verändern. Für die Modellierung der Zeitverzögerungen, verursacht durch das Ab- und Ankoppeln, kann die `IServiceTimePolicy` verwendet werden.

Neben dem Fahrzeug stellt das Metamodell auch ein Strukturelement für den Fahrer (`Driver`) zur Verfügung. Hier beschränken sich die Möglichkeiten der Modellierung auf die Arbeitszeiten des Fahrers. Weitere Restriktionen werden im Simulator nicht berücksichtigt, können jedoch leicht in Form von zusätzlichem unabhängigen Verhalten ergänzt werden.

Das Strukturelement `IOccasionalDriver` ist Bestandteil einer experimentellen Version des RVRPSim. Ein `IOccasionalDriver` modelliert einen unabhängigen Fahrer, der ausgeschriebene Transportaufträge übernehmen kann. Der gelegentliche Fahrer modelliert einen Tagesablauf mithilfe von unabhängigem Verhalten. Über den Tag ändert sich, ausgelöst durch Ereignisse, der Ort und der interne Zustand. Der gelegentliche Fahrer verfügt über ein beliebig komplexes internes Zustandsmodell. Eine Implementierung des Interfaces `IDynamicBehaviourProvider` entscheidet, ob dynamische Anfragen durch die eigene Flotte (Realisierungen der regulären Fahrzeuge und Fahrer) bearbeitet werden sollen oder allen gelegentlichen Fahrern bekannt gemacht werden. Die dynamische Bestellung wird mit einem Bieterverfahren an den niedrigst bietenden Fahrer vergeben und auch von diesem realisiert. Der Fahrer entscheidet abhängig von seinem aktuellen Standort und internen Zustand, bis zu welchem Gebot

die Bestellung umgesetzt werden soll. Mithilfe der experimentellen Version des RVRPSim kann das *VRP with Occasional Drivers* (VRPOD) untersucht werden. Gelegentliche Fahrer sind nicht Bestandteil des Hauptentwicklungszweiges des Simulators und sind nicht veröffentlicht.

Zeitbezogene Restriktionen und Zeitfensterstruktur

Zeitbezogene Restriktionen werden im RVRPSim mithilfe von unabhängigem Verhalten modelliert (vgl. Unterabschnitt 3.2.2). Öffnungszeiten können sowohl am Kunden als auch am Depot existieren. Auf diese Weise ist es möglich Zeitfenster für Bestellungen zu modellieren. Zusätzlich können Zeitfenster auch als Parameter an der Bestellung modelliert werden. Netzwerkelemente verfügen in der aktuellen Version des RVRPSim nicht über eigenes unabhängiges Verhalten. Es können also keine zeitlichen Beschränkungen für Strecken zwischen Kunden direkt modelliert werden. Eine Möglichkeit der Umsetzung ist aber die Implementierung einer `IRoutingPolicy`, die zeitliche Restriktionen von Straßen berücksichtigt.

Inkompatibilitäten und spezifische Restriktionen

Der RVRPSim unterstützt eine Reihe von Inkompatibilitäten. Zum Beispiel, können Güter nur in dafür vorgesehenen Lagern transportiert werden (vgl. Abbildung 3.7). Restriktionen hinsichtlich Fahrzeug und Fahrer können durch eine Implementierung des Planungsalgorithmus berücksichtigt werden. Auch werden teilweise Restriktionen des multimodalen Verkehrs durch den Simulator bereits umgesetzt. Verschiedene Fahrzeuge sind in der Lage Güter mithilfe von abhängigem Verhalten miteinander auszutauschen. Grundsätzlich ist es möglich zusätzliche Restriktionen leicht mithilfe von Richtlinien oder Strategien abzubilden. Es kann aber nicht ausgeschlossen werden, dass die Umsetzung sehr spezifischer Restriktionen größere Anpassungen erfordern.

Zielfunktion

Da nach einer Simulation das gesamte Simulationsmodell zur Verfügung gestellt wird, können verschiedene Statistiken nach Simulationsende ausgewertet werden. Sowohl der Anpassung- als auch der Planungsalgorithmus kann auf das gesamte Simulationsmodell zugreifen und so die unterschiedlichsten Ziele bei der Generierung von Lösungen verfolgen.

In der folgenden Tabelle 3.3 werden die Ergebnisse der vorherigen Diskussion zusammengefasst und der Umsetzungsgrad der Klassen der Taxonomie nach Lahyani et al. (2015) im Simulator aufgezeigt. Die Begrifflichkeit „Vollständig umgesetzt“ wird an dieser Stelle verwendet, wenn notwendige Erweiterungen sich auf die Implementierung von vorhandene Schnittstellen beschränken. Der Umsetzungsgrad wird mit „Teilweise

umgesetzt“ beschrieben, wenn Teile der Klasse der Taxonomie vollständig implementiert sind, andere Teile aber nur mit einer Logikanpassung im Simulator abbildbar sind.

Tabelle 3.3: Klassifikation des RVRPSim nach der Taxonomie von Lahyani et al. (2015)

Klasse nach Lahyani et al. (2015)	Umsetzungsgrad im RVRPSim
Entscheidungsmanagement	Vollständig umgesetzt
Anzahl der Depots	Vollständig umgesetzt
Operationstyp	Vollständig umgesetzt
Ladungsteilung Restriktionen	Vollständig umgesetzt
Planungsperiode	Vollständig umgesetzt
Mehrfachnutzung der Fahrzeuge	Vollständig umgesetzt
Fahrzeuge	Teilweise umgesetzt (Fahrzeug und Anhänger werden nicht getrennt voneinander im Metamodell betrachtet; gelegentliche Fahrer nur in experimenteller Version verfügbar)
Zeitbezogene Restriktionen und Zeitfensterstruktur	Vollständig umgesetzt
Inkompatibilitäten und spezifische Restriktionen	Teilweise umgesetzt (Umsetzung sehr spezifischer Restriktionen erfordert Anpassungen)
Zielfunktion	Teilweise umgesetzt (für sehr spezifische Ziele müssen eventuell weitere Statistiken im Simulationsmodell während der Ausführung erfasst werden)

3.3 Gesamtüberblick und Zusammenfassung

Der in Kapitel 3 vorgestellte RVRPSim legt für diese Arbeit die Grundlage für die Evaluation von verschiedenen Routingalgorithmen für unterschiedliche DVRP-Instanzen (vgl. Abbildung 3.1). Die Implementierung berücksichtigt alle drei Hauptanforderungen, formuliert in der Einleitung in Kapitel 3. Der quelloffene Simulator unterstützt eine Vielzahl von Variationen des VRP, er ist leicht erweiterbar und gut dokumentiert. Das macht den Simulator attraktiv für eine Weiterverwendung über diese Arbeit hinaus.

Im Zuge der Entwicklung des RVRPSim sind eine Reihe weiterer Komponenten, die den Lösungsfindungsprozess für DVRP unterstützten, entwickelt worden. Diese sollen abschließend im Folgenden kurz vorgestellt werden. Alle Komponenten und deren Zusammenwirken sind schematisch in Abbildung 3.15 dargestellt. Ausgangspunkt für den Lösungsfindungsprozess ist die Problemstellung, repräsentiert im REP-VRP-Modellformat, eingeführt von Mendoza et al. (2014). Das Modell unterstützt eine

Vielzahl von Variationen des VRP, kann aber die dynamische Variante des Problems nicht abbilden (vgl. Unterabschnitt 2.1.7). Der entwickelte Generator für dynamische Modelle erweitert ein REP-VRP-Modell hin zu einem dynamischen REP-VRP-Modell. Dazu müssen im Wesentlichen zwei Entscheidungen getroffen werden. Die erste Entscheidung ist, welche Kundenanfragen dynamisch sein sollen. Die zweite ist, wann diese Kundenanfragen bekannt werden. Zum Treffen dieser Entscheidungen verarbeitet der Modellgenerator eine DVRP-Eigenschaftenstrategie, die, zum Beispiel, die Anzahl der dynamischen Kundenanfragen bestimmt und eine Zeitmanagementstrategie, die den Zeitpunkt der dynamischen Anfrage definiert. Der Generator wählt aus den Kundenanfragen zufällig die definierte Menge der dynamischen Anfragen und bestimmt den Zeitpunkt, wann diese Anfrage bekannt werden. Das resultierende Dynamic-REP-VRP-Modell ist eine Erweiterung des REP-VRP-Modells und kann persistiert werden. Das entwickelte Dynamic-REP-VRP-Modell unterstützt aktuell ausschließlich zusätzliche Kundenanfragen. Zu einem Zeitpunkt verschwindende Anfragen werden nicht unterstützt. Das erweiterte REP-VRP-Modell ist ebenfalls unter der Apache-Lizenz 2.0 auf Github veröffentlicht (Mayer, 2018).

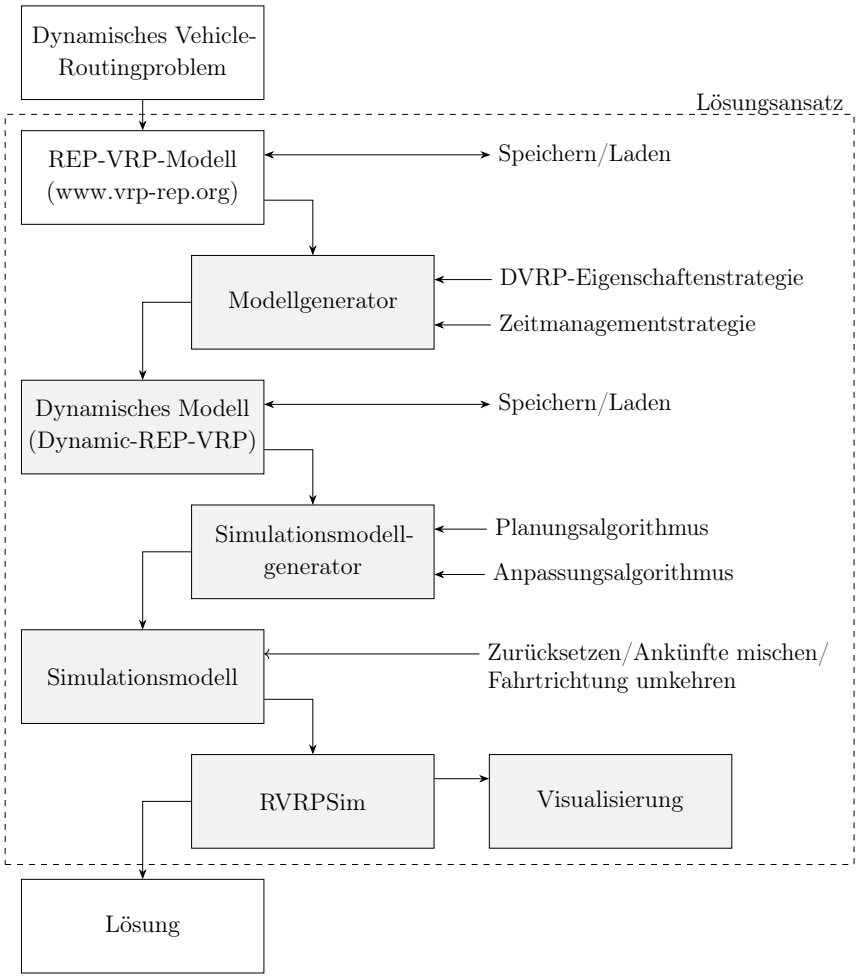

Abbildung 3.15: Darstellung der am Lösungsfindungsprozess beteiligten Komponenten (alle grau markierten Komponenten wurden im Zuge dieser Arbeit entwickelt)

Ein Simulationsmodellgenerator verarbeitet das Dynamic-REP-VRP-Modell zu einem Simulationsmodell. Dazu ordnet der Generator den Elementen des Dynamic-REP-VRP-Modells entsprechende Gegenstücke aus dem Simulationsmetamodell zu. Zusätzlich muss ein Planungsalgorithmus das abhängige Verhalten (vgl. Unterabschnitt 3.2.2) konstruieren. Auch ist es notwendig, einen Anpassungsalgorithmus für das Simulationsmodell zu definieren. Der Algorithmus reagiert während der Simulation auf dynamische Kundenanfragen (vgl. Unterabschnitt 3.2.3). Der implementierte Simulationsmodellgenerator unterstützt aktuell ausschließlich die in dieser Arbeit betrachtete DVRP Variation, kann aber leicht erweitert werden. Das Simulationsmodell verfügt über keine Schnittstelle zum Persistieren, bietet aber Funktionalitäten an, um das Experimentieren zu vereinfachen. So kann ein ausgeführtes Simulationsmodell auf den Startzustand zurückgesetzt werden (`reset`), die Zeitpunkte, wann dynamische Kundenanfragen bekannt werden, können den Anfragen zufällig neu zugeteilt werden (`shuffle`) und die Fahrtrichtung der Touren kann umgekehrt werden (`reverse`). Auf die Notwendigkeit dieser Funktionalitäten wird in Abschnitt 5.1 genauer eingegangen.

Das generierte Simulationsmodell kann jetzt, wie in Unterabschnitt 3.1.2 aufgezeigt, durch den vorgestellten RVRPSim verarbeitet werden. Der Simulator bietet, wie in Abschnitt 3.1 beschrieben, verschiedene Ausführungsmodi an. Die schrittweise Ausführung nutzt die für den Simulator bereitgestellte Visualisierung. Diese erlaubt das vereinfachte Untersuchen von Planungs- und Anpassungsalgorithmen. Abbildung 3.16 zeigt die Visualisierung bei der Darstellung einer Simulation. Die verschiedenen Bereiche sind farblich hervorgehoben. Im blauen Bereich werden alle Kundenanfragen (rote Häuser markieren Standorte von dynamische Kunden), Fahrzeuge und Fahrer mit ihren aktuellen Zuständen dargestellt. Fahrzeug und Fahrer verändern also während der Simulation auf der Visualisierung den Standort. Zusätzlich werden bereits zurückgelegte Strecken (schwarze Pfeile) und zum Simulationszeitpunkt geplante Strecken (hellblaue Pfeile) visualisiert. Mit dem blauen Bereich der Visualisierung kann mit der Maus interagiert werden. Unterstützt werden Zoom mithilfe des Mausrads und Verschieben mithilfe von *Drag and Drop*. Ein Mausklick auf ein Strukturelement visualisiert den aktuellen Zustand des Elements im linken Bereich (rote Markierung) der Visualisierung. Für jedes angeklickte Element entsteht ein Block im linken Bereich, der Informationen wie Lagerbestand oder Ort über das Element enthält. Die Blöcke können mit einem Klick auf das Kreuz in der oberen rechten Ecke geschlossen werden. Durch weitere Blöcke im linken Bereich (rote Markierung) werden die aktuellen Zustände der Touren abgebildet. Diese Blöcke können nicht geschlossen werden.

Im grünen Bereich der Visualisierung werden Informationen über die dynamischen Kundenanfragen bereitgestellt. Diese Informationen erscheinen erst zum Simulationszeitpunkt, an dem der dynamische Kunde bekannt wird. Im gelben Bereich bündeln sich die Optionen zum Steuern der Simulation. Die Simulation kann gestartet, gestoppt oder pausiert werden. Zusätzlich kann die Ablaufgeschwindigkeit konfiguriert werden. Es ist nicht vorgesehen, Simulationsmodelle zu laden, zu speichern oder zu schließen. Das Programm visualisiert genau einen Simulationslauf.

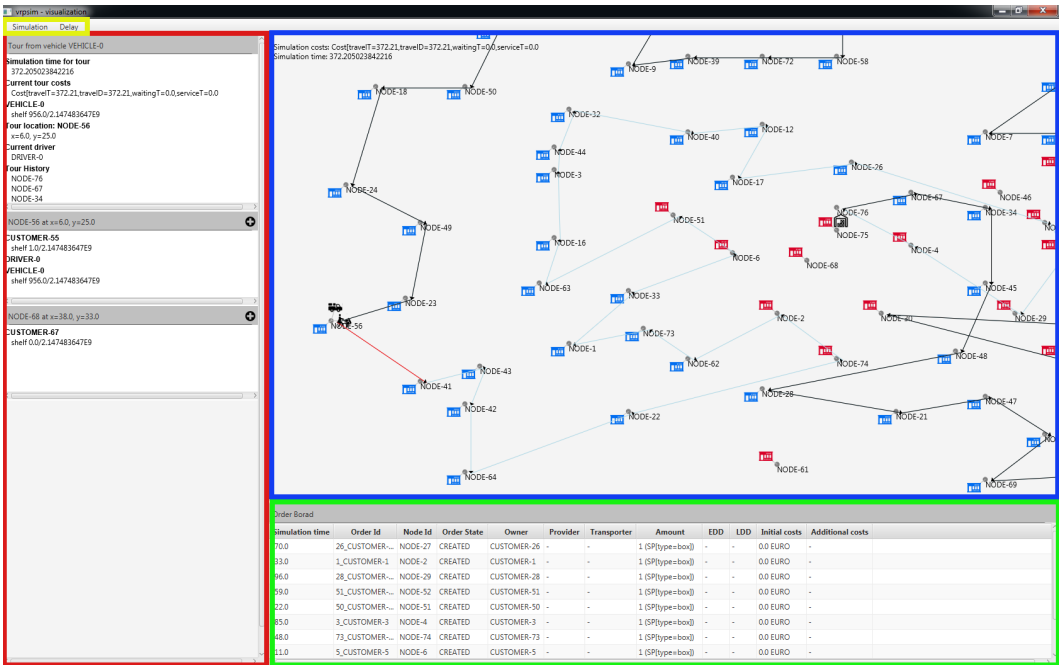

Abbildung 3.16: Darstellung der implementierten Visualisierung für den RVRPSim (verschiedenen Bereiche der Visualisierung sind farblich hervorgehoben)

Alle zusätzlich entwickelten Komponenten sind ebenfalls unter der Apache-Lizenz 2.0 auf Github veröffentlicht (Mayer, 2018).

Kapitel 4

Eigenschaften und Lösungsansätze

Wie in Abbildung 4.1 dargestellt ist, basiert die Auswahl von Planungs- und Anpassungsalgorithmen durch einen Optimierer auf einer Beschreibung der DVRP-Instanzen mithilfe von Eigenschaften. Eine Algorithmenauswahl auf der Grundlage der gesamten Probleminstanz ist aufgrund der in Unterabschnitt 2.2.1 genannten Gründe nicht zielführend. Problemeigenschaften ermöglichen das Gruppieren von Probleminstanzen anhand von Gemeinsamkeiten und Unterschieden. Erst die Unterscheidbarkeit von Instanzen ermöglicht einen fairen Vergleich von Planungs- und Anpassungsalgorithmen. Das ASP (Rice, 1976) und die No-Free-Lunch-Theoreme (Macready und Wolpert, 1996) zeigen, dass die Performance von Algorithmen abhängig von den Problemeigenschaften ist (vgl. Abschnitt 2.2). Leicht kann bei der ungeeigneten Auswahl von Probleminstanzen für einen Algorithmenvergleich der Eindruck entstehen, dass ein Algorithmus einem anderen ganz allgemein überlegen ist. Die Bewertung der Problemstellungen mithilfe von Eigenschaften ist aus diesem Grund für einen fairen Vergleich von Algorithmen essenziell. Zudem erhöht die Unterscheidbarkeit von Instanzen das Verständnis über das Verhalten von Lösungsansätzen.

Für das DVRP existieren nur wenige Eigenschaften, anhand derer sich Problemstellungen voneinander abgrenzen lassen (vgl. Unterabschnitt 2.1.3). Für die Umsetzung des in Abbildung 4.1 dargestellten Ansatzes müssen Problemeigenschaften für das DVRP eingeführt werden. Die Arbeit greift dabei auf bekannte Problemeigenschaften für das TSP zurück. Eine detaillierte Übersicht dieser Eigenschaften ist, zum Beispiel, in Unterabschnitt 2.2.2 zu finden. Bevor in Unterabschnitt 4.1.2 und 4.1.3 Problemeigenschaften für das DVRP eingeführt werden, erläutert Unterabschnitt 4.1.1 Ansätze für die Evaluation der Eigenschaften unabhängig von Algorithmenselektion.

Wie in Abbildung 4.1 angedeutet ist, richtet sich der zweite Fokus des Kapitels auf die Planungs- und Anpassungsalgorithmen. Abschnitt 4.2 beschäftigt sich dementsprechend mit umgesetzten Lösungsansätzen. Es werden etablierte Verfahren, aber auch aktuelle Ansätze vorgestellt, mit denen die dynamischen Problemstellungen gelöst werden. Zusätzlich wird der im Zuge dieser Arbeit entwickelte Planungsalgorithmus vorgestellt. Der Algorithmus erweist sich als sehr erfolgreich für Probleminstanzen mit bestimmten Ausprägungen von Eigenschaften. Alle vorgestellten Lösungsansätze stehen im bereits

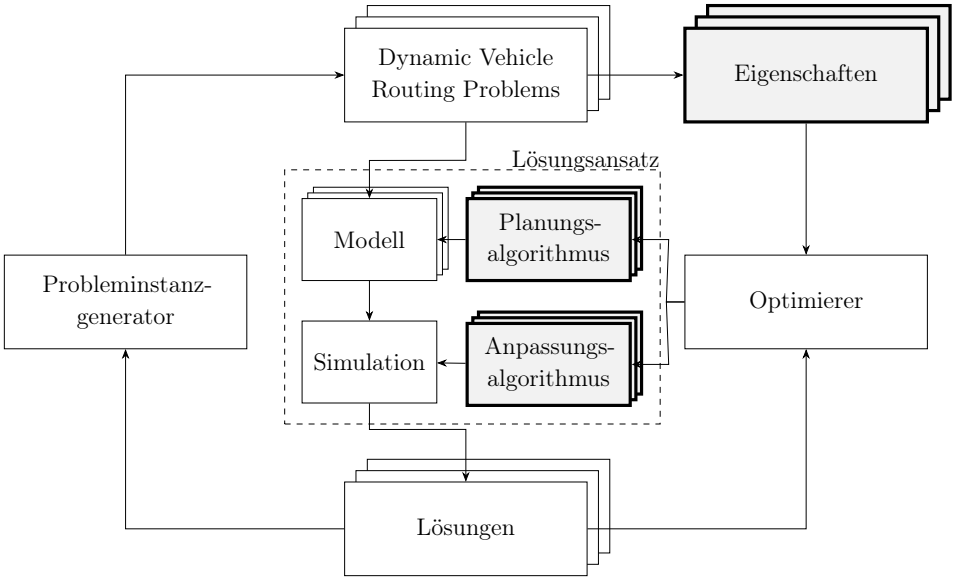

Abbildung 4.1: Lösungsansatz mit Fokus auf Eigenschaften sowie Planungs- und Anpassungsalgorithmus

beschriebenen *Rich Vehicle Routing Problem Simulator* (RVRPSim) zur Verfügung (vgl. Kapitel 3). Abschnitt 4.3 schließt das Kapitel mit einer Zusammenfassung.

4.1 Eigenschaften Dynamischer Routingprobleme

Für das DVRP werden in der Literatur nur wenige, die Dynamik beschreibende intrinsische Problem- bzw. Instanzeigenschaften betrachtet (vgl. Unterabschnitt 2.1.3). Aktuell können demzufolge Problemstellungen nur rudimentär anhand ihrer Dynamik voneinander abgegrenzt werden. Die Unterscheidbarkeit von Problemen ist aber die Grundvoraussetzung für einen fairen Vergleich von Algorithmen. Mithilfe von Problemeigenschaften kann sichergestellt werde, dass Testinstanzen signifikant unterschiedlich sind und diese eine aussagekräftige Analyse des Verhaltens von Algorithmen erlauben. Auch für die automatisierte Auswahl von Algorithmen sind Problemeigenschaften unverzichtbar (vgl. Abbildung 2.9). Aus diesen Gründen werden im nun folgenden Unterkapitel intrinsische Problemeigenschaften für das DVRP eingeführt. Als Ausgangspunkt für die Entwicklung von Eigenschaften dient das vom Autor dieser Arbeit in Mayer et al. (2017) eingeführte Lemma 4.1.1. Mayer et al. (2017) identifizieren die Eigenschaften Ort und Zeit als fundamentale Problemeigenschaft einer dynamischen Kundenanfrage.

Lemma 4.1.1. *Eine dynamische Kundenanfrage in einem DVRP muss mindestens mit den Eigenschaften* ***Zeit*** *und* ***Ort*** *modelliert werden.* ***Zeit*** *und* ***Ort*** *sind demzufolge fundamentale Problemeigenschaften einer dynamischen Kundenanfrage, die unabhängig von einer konkreten Problemstellung existieren.*

Die vorliegende Arbeit konzentriert sich auf die in Lemma 4.1.1 definierte fundamentale Eigenschaft Ort eines DVRP. Diese ist für alle Variationen des DVRP berechenbar. Den Zeitpunkt, an dem eine dynamische Anfrage einem Planer bekannt wird, berücksichtigt der von Larsen (2000) eingeführte EDOD (vgl. Abschnitt 2.1.3). Aus diesem Grund wird die Eigenschaft Zeit für die Erarbeitung neuer Problemeigenschaften in der vorliegenden Arbeit nicht näher betrachtet. In Unterabschnitt 2.1.3, in Tabelle 2.2 wird die Eigenschaft Ort, der REP-VRP-Modellnotation folgend, als Knoten bezeichnet.

Für das TSP existieren verschiedene ortsbezogene Problemeigenschaften (vgl. Unterabschnitt 2.2.2). Diese können auch, in angepasster Form, DVRP-Instanzen charakterisieren. Die hier eingeführten, die Dynamik beschreibenden Problemeigenschaften werden in zwei Kategorien eingeteilt. Die erste Kategorie umfasst Problemeigenschaften, die ausschließlich dynamische Kundenanfragen charakterisieren. Hier werden bekannte TSP-Eigenschaften auf die Menge der dynamischen Kundenanfragen angewendet. Eigenschaften dieser Kategorie werden in Unterabschnitt 4.1.2 eingeführt. Die zweite Kategorie umfasst Eigenschaften, die die Beziehungen zwischen dynamischen und statischen Kundenanfragen beschreiben. Ein bekannter Vertreter der zweiten Kategorie ist, zum Beispiel, der DOD, eingeführt von Lund et al. (1996). Der DOD beschreibt das Verhältnis zwischen der Anzahl von dynamischen zu statischen Kundenanfragen (vgl. Abschnitt 2.1.3). Die Eigenschaften der zweiten Kategorie werden in Unterabschnitt 4.1.3 vorgestellt. Alle eingeführten Eigenschaften werden nach dem betrachteten Charakteristikum mit dem Zusatz „-Eigenschaft“ benannt. Zusätzlich wird der Name der Eigenschaft um den Namen der berechneten Statistik ergänzt. So wird, zum Beispiel, bei der eingeführten Eigenschaft *Winkel-Eigenschaft Durchschnitt* die Statistik *Durchschnitt* über das Charakteristikum *Winkel* betrachtet. Für Eigenschaften der zweiten Kategorie werden teilweise Statistiken über Statistiken berechnet. Diese zusätzliche berechnete Statistik wird ebenfalls dem Namen der Eigenschaft hinzugefügt. So wird, zum Beispiel, für die Eigenschaft *Winkel-Eigenschaft Durchschnitt-Durchschnitt* ein Durchschnitt über alle Durchschnitte der Winkel für jede dynamische Anfrage einer Instanz bestimmt. Zwei verschiedene Ansätze für die Evaluation der Wertebereiche der eingeführten Eigenschaften werden in Unterabschnitt 4.1.1 diskutiert.

4.1.1 Grundlagen für die Evaluation der Eigenschaften

Für die Analyse und Evaluation der eingeführten DVRP-Eigenschaften werden zwei unterschiedliche Ansätze verfolgt. Für den ersten Ansatz werden verschiedene Typen von Probleminstanzen definiert. Diese Instanzen dienen als Grundlage für die Analyse der Wertebereiche der DVRP-Eigenschaften. Es wird untersucht, ob sich die Instanzen anhand der eingeführten Eigenschaften einem Typ zuordnen lassen. Wenn diese Zuordnung möglich ist, ist grundsätzlich gezeigt, dass sich die eingeführten Problemeigenschaften zum Unterscheiden von DVRP-Instanzen eignen. Die Instanztypen dienen zusätzlich als Grundlage für Rückschlüsse über die Aussagekraft der Eigenschaften auf die Beschaffenheit der Probleminstanzen. Die Generierung der verschiedenen Typen wird im folgenden

Abschnitt 4.1.1 diskutiert. Der zweite Ansatz für die Analyse der Eigenschaften versucht Probleminstanzen zu erzeugen, bei denen die eingeführte Eigenschaft minimal bzw. maximal ausgeprägt ist. Mithilfe der Visualisierungen der Instanzen sollen die gezogenen Rückschlüsse zwischen Eigenschaften und Beschaffenheit einer Probleminstanz validiert werden. In Unterabschnitt 5.2.1 werden die Problemeigenschaften identifiziert, die einen Einfluss auf die Lösungsqualität von Algorithmen haben. Die Visualisierungen aus diesem Kapitel geben einem Anwender einen Eindruck von der Ausprägung von Problemeigenschaften. So kann auch unabhängig von automatisierten Verfahren aufgrund von Ähnlichkeiten von Probleminstanzen auf Merkmalsausprägungen geschlossen und eine manuelle Selektion von Algorithmen durchgeführt werden. Der für die Erzeugung der Instanzen implementierte Algorithmus wird in einem der folgenden Unterkapitel vorgestellt.

Generierung von unterschiedlichen Instanztypen

Die Wertebereiche der in den folgenden Abschnitten eingeführten DVRP-Eigenschaften werden mithilfe von Probleminstanzen unterschiedlichen Typs untersucht. Abbildung 4.2 zeigt die vier betrachteten Typen von Probleminstanzen, die sich an typischen Clusteringbenchmarks orientieren, siehe, zum Beispiel, Fränti und Sieranoja (2018). Ein Typ umfasst 100 Kundenanfragen auf einem $r \times r$ Raster mit $r = 100$. Ein DOD von 0,5 definiert die Anzahl der dynamischen Anfragen pro Instanz. Der Zeitpunkt t_i für eine dynamische Anfrage i wird so gewählt, dass alle Zeitpunkte äquidistant im betrachteten Zeitraum verteilt sind. Der betrachtete Zeitraum wird durch den Zeithorizont T begrenzt. Die Generierung der Zeitpunkte t_i erfolgt nach der folgenden Regel: $t_{i+1} - t_i = d_i; d_i = d_{i+1} \forall i$, wobei $t_{n+1} = T$. T wird mithilfe des 2-Opt-Algorithmus bestimmt, der in Abschnitt 4.2 genauer vorgestellt wird. Das Depot aller erzeugten Instanzen befinden sich bei der Rasterkoordinate $(\frac{r}{2},\frac{r}{2})$.

In Probleminstanzen vom Typ A können dynamische und statische Anfragen linear voneinander getrennt werden. Die X-Koordinaten der dynamischen Anfragen sind im Intervall $[0,\frac{r}{2}]$ gleichverteilt. Die X-Koordinaten der statischen Anfragen sind im Intervall $(\frac{r}{2},r]$ gleichverteilt. Die Y-Koordinaten der Anfragen in Probleminstanzen von Typ A sind im Intervall $[0,r]$ gleichverteilt. In Instanzen von Typ B umschließen die statischen die dynamischen Anfragen. Die Koordinaten der statischen und dynamischen Anfragen werden mithilfe einer Verwerfungsmethode bestimmt. Die X- und Y-Koordinaten werden im Intervall $[0,r]$ gleichverteilt erzeugt. Befindet sich der generierte Punkt innerhalb des Kreises K_1 mit einem Radius von $\frac{r}{3}$ und dem Mittelpunkt in der Rasterkoordinate $(\frac{r}{2},\frac{r}{2})$, so handelt es sich um eine dynamische Anfrage. Befindet sich der erzeugte Punkt außerhalb des Kreises K_2 mit einem Radius von $\frac{r}{3} + \frac{r}{10}$ und dem Mittelpunkt in der Koordinate $(\frac{r}{2},\frac{r}{2})$, so handelt es sich um eine statische Anfrage. In Probleminstanzen vom Typ C umschließen die dynamischen die statischen Anfragen. Auch hier werden die Anfragen mithilfe einer Verwerfungsmethode generiert. Erzeugte Anfragen, die sich innerhalb des Kreises K_1 befinden sind statische Anfragen, wohingegen dynamische Anfragen außerhalb von K_2 liegen. Typ D charakterisiert Instanzen, bei denen dy-

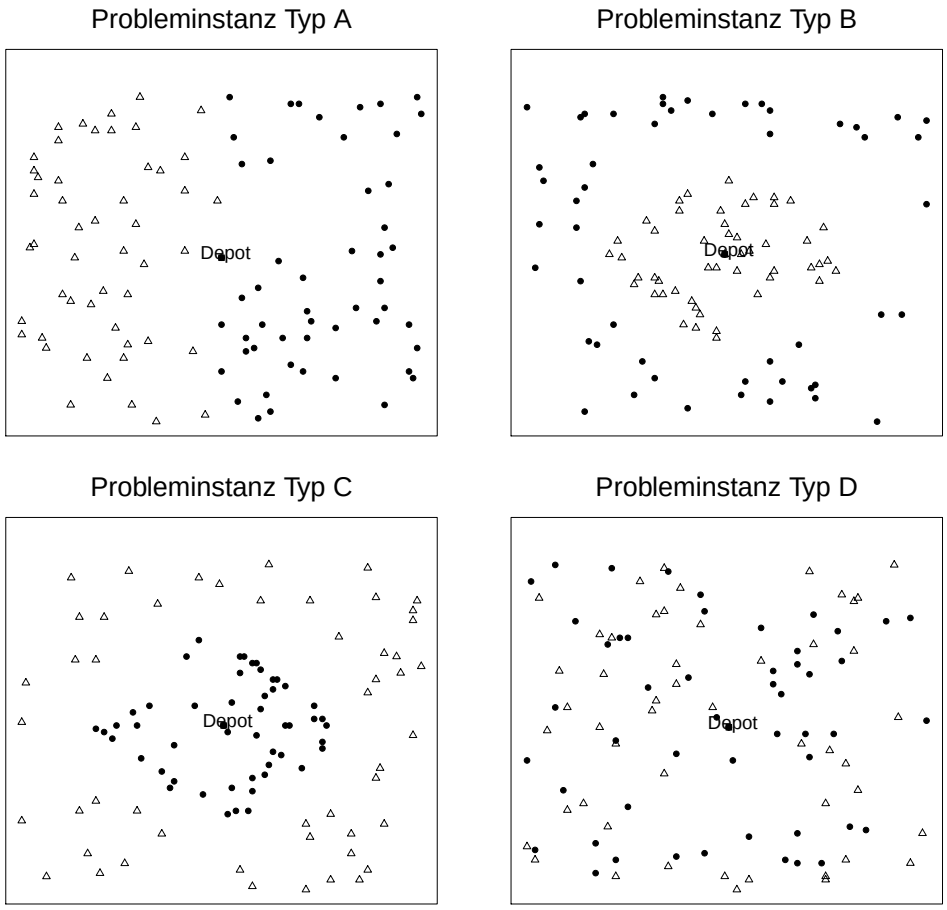

Abbildung 4.2: Typen von DVRP-Instanzen erzeugt für die Evaluierung der eingeführten DVRP-Eigenschaften

namische und statische Anfragen unabhängig voneinander verteilt sind. Die X- und Y-Koordinaten, sowohl der dynamischen, als auch der statischen Anfragen, sind im Intervall $[0,r]$ gleichverteilt.

Für jeden Typ werden 1.000 zufällige Instanzen erzeugt. Die ermittelten Werte der DVRP-Eigenschaften für alle erzeugten Instanzen bilden die Grundlage für die Analyse der Wertebereiche der Eigenschaften.

Instanzen mit minimalen und maximalen Ausprägungen der Eigenschaften

Um Korrelationen zwischen der Beschaffenheit von Probleminstanzen und Eigenschaften auch visuell nachvollziehen zu können, ist es hilfreich, Instanzen zu generieren, bei denen die Ausprägungen der Eigenschaften möglichst minimal bzw. maximal sind. Durch die Visualisierung der generierten Instanzen wird es möglich, vermutete Zusammenhänge zu bestätigen, oder zu verwerfen. Ein einfacher, valider Ansatz zum Ermitteln solcher Probleminstanzen ist, zum Beispiel, die Brute-Force-Methode. Es werden sehr viele zufällige Instanzen generiert und die mit der größten Differenz der Ausprägungen der Eigenschaften visualisiert. Der große Nachteil dieser Methode ist das zufällige

Erzeugen von Instanzen. Das Histogramm in Abbildung 4.3 zeigt die Verteilung der Werte der Distanz-Eigenschaft Durchschnitt (vgl. Abschnitt 4.1.2) bei 1.000 zufällig erzeugten Probleminstanzen nach Typ D. Gut ist zu sehen, dass die Werte annähernd einer Normalverteilung folgen. Auch die anderen eingeführten Eigenschaften zeigen dieses Verhalten. Eine zufällig erzeugte Probleminstanz wird also wahrscheinlich eine Ausprägung der Eigenschaften um den Mittelwert der Verteilungen haben. Die für diese Analyse benötigten Extreme, Instanzen an den Rändern der Normalverteilung, werden zufällig wesentlich seltener erzeugt. Aus diesem Grund erweist sich die Brute-Force-Methode als sehr aufwendig und ist damit ungeeignet.

Ein effizienterer Ansatz für die Generierung von Probleminstanzen mit einer möglichst maximalen Differenz der Ausprägungen der Eigenschaften ist die gezielte Suche nach solchen Instanzen. Dabei kommt ein Optimierungsverfahren zum Einsatz, das über die Manipulation der Probleminstanz die Ausprägung der Eigenschaft maximiert bzw. minimiert. Für die Umsetzung der Suche wird ein Genetischer Algorithmus (GA) auf Basis des Meta-Heuristik-Frameworks SEREIN, eingeführt in Uhlig (2015), implementiert. Ausgangspunkt für den Algorithmus ist die in Abbildung 4.3 visualisierte, einfache VRP-Instanz. Der Freiheitsgrad des Algorithmus bei der Manipulation der Probleminstanz beschränkt sich auf die Auswahl von $n_{dynamisch}$ dynamischen Anfragen aus allen vorhandenen Kundenanfragen. Auf einen größeren Freiheitsgrad, zum Beispiel, das freie Platzieren der Anfragen, wird aufgrund von der aufwendigen Definition geeigneter Grenzwerte verzichtet. Ein DOD von 0,5 definiert die Anzahl $n_{dynamisch}$ der dynamischen Anfragen pro Instanz.

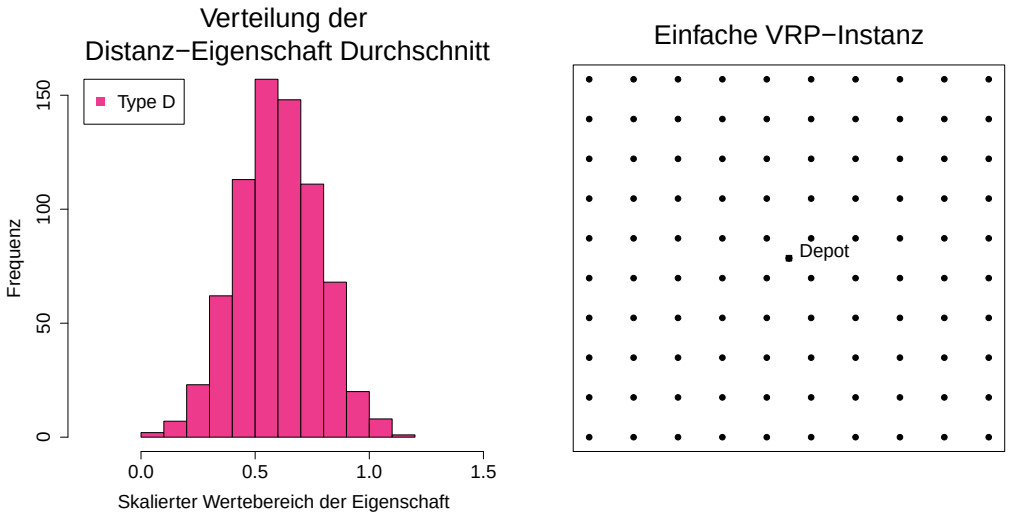

Abbildung 4.3: Histogramm über die Verteilung der Distanz-Eigenschaft Durchschnitt (vgl. Abschnitt 4.1.2) und einfache VRP-Instanz (Ausgangspunkt für die heuristische Suche nach dynamischen Probleminstanzen)

Bei dem gegebenen Freiheitsgrad kann die Hauptaufgabe des GA auf die Sortierung aller Kundenanfragen abstrahiert werden. Die folgende Aufzählung listet die implementierten Komponenten des Algorithmus mit kurzen Erläuterungen.

- **GA-Modell der Probleminstanz** - Die DVRP-Instanz wird mithilfe einer Liste von Integerwerten repräsentiert. Die Werte entsprechen den Identifikatoren der Kundenanfrage. Mithilfe der Identifikatoren kann ein Bezug zur X-/Y-Koordinate der Anfrage hergestellt werden. Ein Modell entspricht einem Individuum.
- **Mutationsoperator** - Der Operator vertauscht Werte im GA-Modell. Es werden ausschließlich dynamische mit statischen Anfragen vertauscht. Die ersten $n_{dynamisch}$ Anfragen im GA-Modell repräsentieren dynamische Anfragen.
- **Rekombinationsoperator** - Der Operator kombiniert zwei GA-Modelle. Alle dynamischen Anfragen, die in beiden Modellen vorkommen, werden in das neue Modell übernommen. Die verbleibende Anzahl dynamischer Anfragen werden zufällig aus den dynamischen Anfragen der Modelle ausgewählt.
- **Fitnessfunktion** - Die Funktion übersetzt das GA-Modell in eine DVRP-Instanz und berechnet die Ausprägung der untersuchten Eigenschaft. Dieser Wert entspricht der Fitness des GA-Modells.

Der GA wird mit zufälligen Modellen der Probleminstanz initialisiert. Diese ersten Individuen bilden die Startpopulation. SEREIN bewertet die Individuen mithilfe der Fitnessfunktion. Die besten Individuen rekombiniert oder mutiert SEREIN und bildet so eine Folgepopulation. Für jede eingeführte DVRP-Eigenschaft sucht SEREIN das Minimum und das Maximum der Fitnessfunktion. Dabei werden 40 Populationen im Verlauf der Suche erzeugt, wobei eine Population aus 15 Individuen besteht. Die gefundenen Instanzen, mit nahe minimaler bzw. maximaler Fitnessfunktion, bilden die Grundlage für die visuelle Analyse der Problemeigenschaften. Grundsätzlich gilt, dass eine heuristische Suche nicht zuverlässig ein Minimum bzw. Maximum finden kann, sondern sich diesen optimalen Ausprägungen der Eigenschaft nur annähert. Ergebnisse nahe dem realen Minimum bzw. Maximum sind hier aber für das Nachvollziehen vermuteter Zusammenhänge zwischen Beschaffenheit einer Probleminstanz und Eigenschaft durchaus ausreichend.

4.1.2 Eigenschaften der dynamischen Anfragen

Die im folgenden Abschnitt eingeführten DVRP-Eigenschaften konzentrieren sich ausschließlich auf die dynamischen Kundenanfragen einer Problemstellung. Dabei wird eine Teilmenge der bekannten TSP-Eigenschaften, vorgestellt in Unterabschnitt 2.2.2, für die Menge der dynamischen Anfragen einer Probleminstanz berechnet. Nachfolgend werden die Eigenschaften mit den zugehörigen Untersuchungen vorgestellt.

Der RVRPSim stellt eine Bibliothek zur Berechnung aller Problemeigenschaften für ein dynamisches REP-VRP-Modell bereit (vgl. Abschnitt 3.3). Die Berechnungen greifen auf das R-Paket tspmeta, eingeführt von Mersmann et al. (2013), zurück.

Winkel-Eigenschaften WE_{dyn}

Alle Winkel-Eigenschaften WE_{dyn} berechnen Statistiken über den Schnittwinkel α zwischen einer dynamischen Kundenanfrage v_i und seinen zwei nächsten dynamischen Anfragen v_k und v_m (vgl. Abbildung 4.4). Für jede Probleminstanz werden alle Schnittwinkel α für alle dynamischen Anfragen $n_{dynamisch}$ bestimmt. Der Durchschnitt und der Variationskoeffizient aller berechneten Schnittwinkel werden als Winkel-Eigenschaften WE_{dyn} für dynamische Kundenanfragen eingeführt. Häufig werden weitere Statistiken über die Schnittwinkel als TSP-Eigenschaften herangezogen. Die Berechnungsvorschriften für den Schnittwinkel α und den Variationskoeffizient $VarK$ sind in Abschnitt A.1 zusammengefasst.

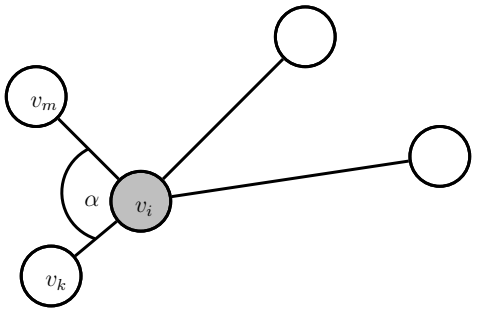

Abbildung 4.4: Schnittwinkel α zwischen einer dynamischen Kundenanfrage v_i und seinen zwei nächsten dynamischen Anfragen v_k und v_m

Abbildung 4.5 zeigt die Ergebnisse der Analyse der Winkel-Eigenschaften WE_{dyn} mit den in Abschnitt 4.1 eingeführten Probleminstanzen. Die berechneten Winkel-Eigenschaften für die Instanzen sind farblich unterschieden nach Typ der Instanz in einem Histogramm abgebildet. Gut ist zu sehen, dass sich die Typen nicht klar anhand ihrer Wertebereiche voneinander abgrenzen lassen. Die Histogramme lassen für alle Typen auf eine Normalverteilung mit ähnlichem Mittelwert und Standardabweichung schließen. Sowohl der Durchschnitt als auch der Variationskoeffizient der Schnittwinkel eignen sich unabhängig von weiteren Problemeigenschaften nicht, um die in Abbildung 4.2 abgebildeten Typen von Instanzen voneinander zu unterscheiden. Auch die Kombination der beiden Eigenschaften erlaubt keine eindeutige Identifizierung. Lediglich für die Unterscheidung von Typ B und D gibt die Kombination der Winkel-Eigenschaften einen ersten Hinweis. So scheint der Durchschnitt der Schnittwinkel für Instanzen vom Typ B im Mittel leicht größer zu sein, als für Instanzen vom Typ C. Das Mittel der Variationskoeffizient der Schnittwinkel für Instanzen vom Typ C scheint dagegen geringfügig größer zu ein als die für Typ B. Beobachtet man für zwei unbekannte Probleminstanzen ein solches Verhalten der Winkel-Eigenschaften, kann das ein schwacher Hinweis für die Zuordnung zu den Typen B und C sein.

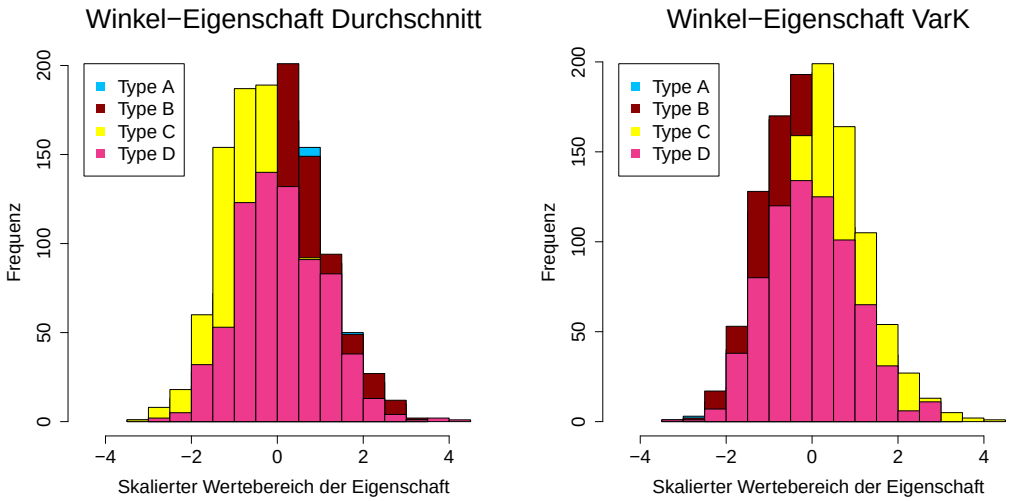

Abbildung 4.5: Wertebereiche der Winkel-Eigenschaften Durchschnitt und Variationskoeffizient $VarK$ nach Typen von Probleminstanzen (vgl. Abbildung 4.2)

Abbildung 4.6 zeigt die DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Werte für die Winkel-Eigenschaften WE_{dyn} Durchschnitt und Variationskoeffizient. Gut ist zu sehen, dass die Probleminstanz mit einem hohen Wert für die Winkel-Eigenschaft Durchschnitt eher Ketten von dynamischen Kundenanfragen aufweist. In einer Kette sind die Schnittwinkel α zwischen Anfragen mit 180° maximal. Probleminstanzen mit niedrigen Werten der Schnittwinkel weisen eher Verbünde von Anfragen auf. Ein hoher Variationskoeffizient der Winkel mischt Ketten und Verbünde, wohingegen ein geringerer Wert des Variationskoeffizient auch wieder zu Verbünden von dynamischen Anfragen neigt. Obwohl sich die Winkel-Eigenschaften nicht zum Unterscheiden der Instanztypen aus Abschnitt 4.1.1 eignen, werden die Eigenschaften weiterhin berücksichtigt. Es kann nicht ausgeschlossen werden, dass weitere Typen von Instanzen existieren, die auf der Grundlage der Winkel-Eigenschaften unterschieden werden können.

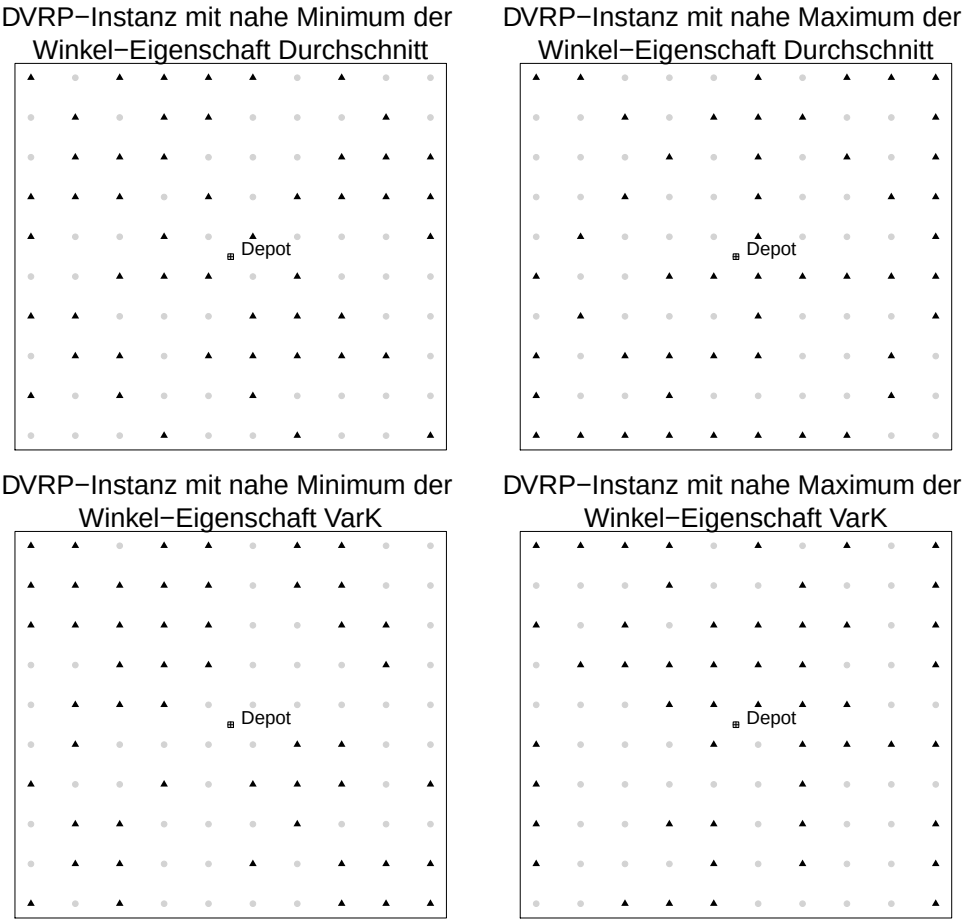

Abbildung 4.6: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Winkel-Eigenschaften Durchschnitt und Variationskoeffizient

Distanz-Eigenschaften DE_{dyn}

Die Distanz-Eigenschaften DE_{dyn} berechnen Statistiken zu den Distanzen (Kosten $c_{i,j}$) von einer dynamischen Kundenanfrage v_i zu allen anderen dynamischen Anfragen v_j (vgl. Abbildung 4.7). Der Durchschnitt und der Variationskoeffizient berechnet aus den Kosten $c_{i,j}$ für alle Anfragen v_i werden als Distanz-Eigenschaften eingeführt.

Die Ergebnisse der Analyse der Distanz-Eigenschaften mithilfe der verschiedenen Typen von DVRP-Instanzen ist in Abbildung 4.8 abgebildet. Gut ist für die Distanz-Eigenschaft Durchschnitt zu sehen, dass sich die unterschiedlichen Typen von Instanzen gut unterscheiden lassen. Die Wertebereiche weisen kaum Überschneidungen auf. Für Instanzen vom Typ B sind die Durchschnittskosten im Vergleich zu den Durchschnittskosten der anderen Typen am niedrigsten. Alle dynamischen Anfragen sammeln sich hier im Zentrum der Probleminstanz und befinden sich so relativ nah beieinander. Für Problemeigenschaften vom Typ C sind die Durchschnittskosten am größten. Alle dynamischen Anfragen sind in einer solchen Instanz eher am Rand zu finden und weisen damit einen großen Abstand untereinander auf (vgl. Abbildung 4.2). In Instanzen vom Typ D sind die dynamischen Anfragen über den gesamten Problemraum gleichverteilt.

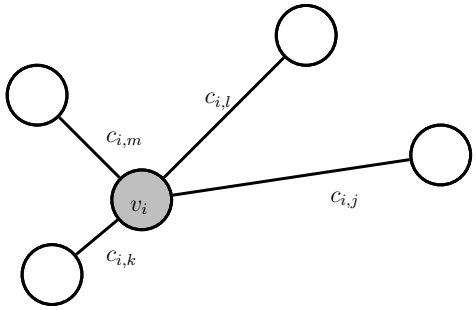

Abbildung 4.7: Dynamische Kundenanfrage v_i mit Distanzen $c_{i,j}$, $c_{i,k}$, $c_{i,l}$ und $c_{i,m}$ zu allen anderen dynamischen Kundenanfragen

Die durchschnittlichen Distanzen sind erwartungsgemäß größer als die durchschnittlichen Distanzen von Instanzen vom Typ A, bei denen dynamische Anfragen ausschließlich auf einer Hälfte des Problemraums zu finden sind (vgl. Abbildung 4.2). Trotz der guten Separierbarkeit der Instanztypen lässt sich leicht nachvollziehen, dass die Distanz-Eigenschaft Durchschnitt nur begrenzte Aussagekraft hat. Die durchschnittlichen Kosten hängen stark von der räumlichen Beziehung der dynamischen Anfragen untereinander ab. Kundenanfragen, die weit voneinander entfernt sind, weisen hohe durchschnittliche Kosten auf. Liegen die Anfragen hingegen eng beieinander sind die Kosten vermutlich niedrig. Der Wert der Eigenschaft erlaubt keine Interpretation über die Lage der Anfragen im Problemraum. Wenn, zum Beispiel, alle dynamischen Kundenfragen von Instanzen vom Typ B einen ähnlich großen Raum einnehmen wie Anfragen von Instanzen vom Typ A, wären die beiden Typen nicht mehr anhand der Distanz-Eigenschaft Durchschnitt unterscheidbar.

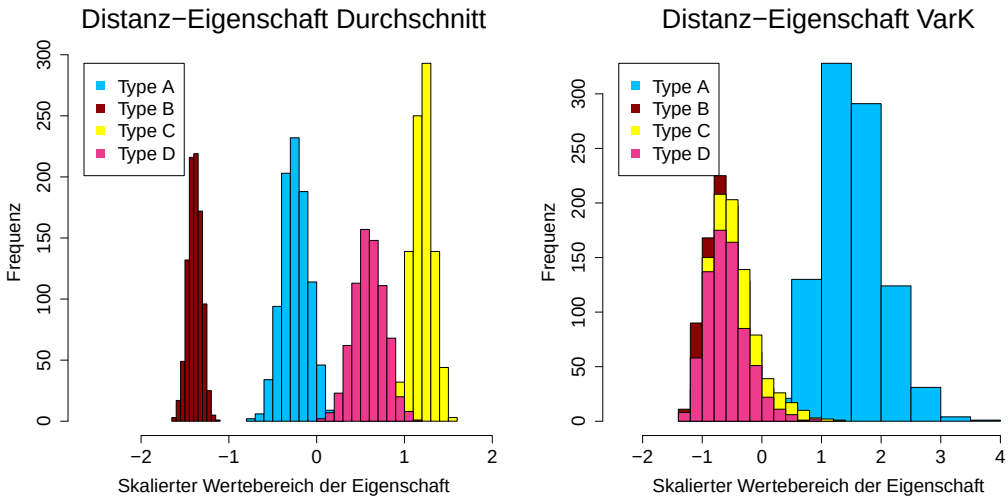

Abbildung 4.8: Wertebereiche der Distanz-Eigenschaften Durchschnitt und Variationskoeffizient $VarK$ nach Typen von Probleminstanzen (vgl. Abbildung 4.2)

Für die Variationskoeffizient der Distanzen der dynamischen Kundenanfrage zeigen die Werte für Instanzen vom Typ B, C und D fast identische Verteilungen. Ausschließlich bei

Instanzen vom Typ A ist das relative Streuungsmaß größer. Eine weitere Untersuchung der Distanz-Eigenschaft Variationskoeffizient zeigt, dass die Streuung der Distanzen mit dem Betrag der Differenz von Höhe und Breite der Fläche auf denen die dynamischen Anfragen verteilt sind korreliert. Abbildung 4.9 zeigt zwei weitere Variationen der Instanzen vom Typ A. Bei Instanzen vom Typ A 30% können dynamische und statische Anfragen auch linear voneinander getrennt werden. Die X-Koordinate wird aber nur im Intervall von $(\frac{N}{3},N]$ gleichverteilt. Die Y-Koordinaten der Anfragen ist weiterhin im Intervall $[0,N]$ gleichverteilt. Damit wird die Differenz der Höhe und Breite der Fläche, in denen die dynamischen Anfragen verteilt sind, größer im Vergleich zu Instanzen vom Typ A. Für Instanzen vom Typ A 20% ist die Differenz von Höhe und Breite der Fläche noch größer. Die X-Koordinate wird hier nur im Intervall von $(\frac{N}{5},N]$ gleichverteilt. Das Histogramm in Abbildung 4.9 bestätigt, dass sich mit zunehmendem Betrag der Differenz von Höhe und Breite der Fläche, auf denen die dynamischen Anfragen verteilt sind, der Mittelwert der Variationskoeffizient erhöht.

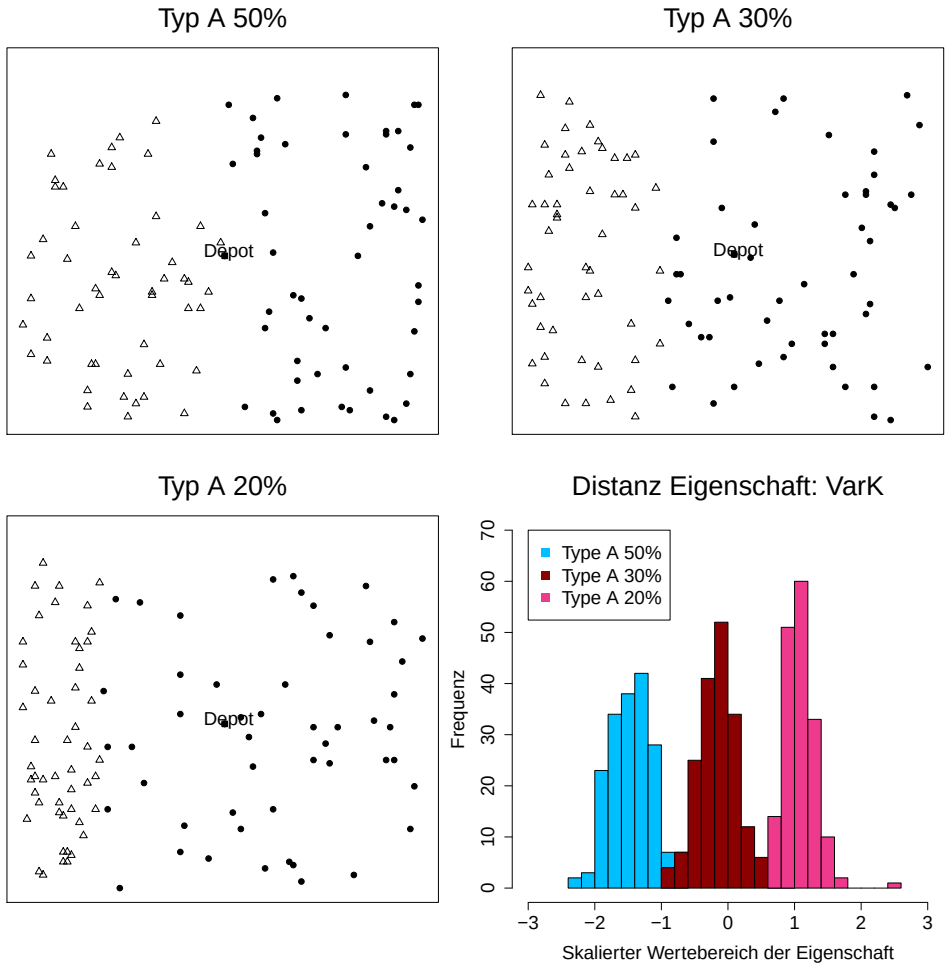

Abbildung 4.9: Erzeugte DVRP-Instanzen für die weitere Untersuchung der Distanz-Eigenschaft Variationskoeffizient und Histogramm (visualisiert Zusammenhang zwischen Betrag der Differenz von Höhe und Breite der Fläche und Distanz-Eigenschaft Variationskoeffizient)

Abbildung 4.10 zeigt die DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Werte für die Distanz-Eigenschaften Durchschnitt und Variationskoeffizient. In Probleminstanzen mit hohen durchschnittlichen Distanzen zwischen dynamischen Anfragen verteilen sich die Kundenanfragen am Rand der Instanz. Bei niedrigen Werten bündeln sich die dynamischen Anfragen dicht beieinander und werden von den statischen Anfragen umschlossen. Die generierten Instanzen ähneln stark den Instanzen der Typen C und B (vgl. Abbildung 4.2). Die Probleminstanz mit einem großen Wert für die Distanz-Eigenschaft Variationskoeffizient zeigt zwei separierte Gruppen von dynamischen Anfragen. Eine sehr große und eine eher kleine Gruppe, die maximal voneinander entfernt sind. Hier gibt es also wenige dynamische Anfragen, die weit vom Rest der anderen dynamischen Anfragen entfernt sind, und viele Anfragen, die nah zueinander sind. Der Variationskoeffizient der Distanz ist ein guter Indikator, ob neben einer großen Ansammlung von dynamischen Anfragen auch kleinere, weiter entfernte Gruppen von Anfragen existieren. In der Instanz, die einen kleinen Wert für den Variationskoeffizienten aufweist, sind die dynamischen Anfragen eher gleichverteilt.

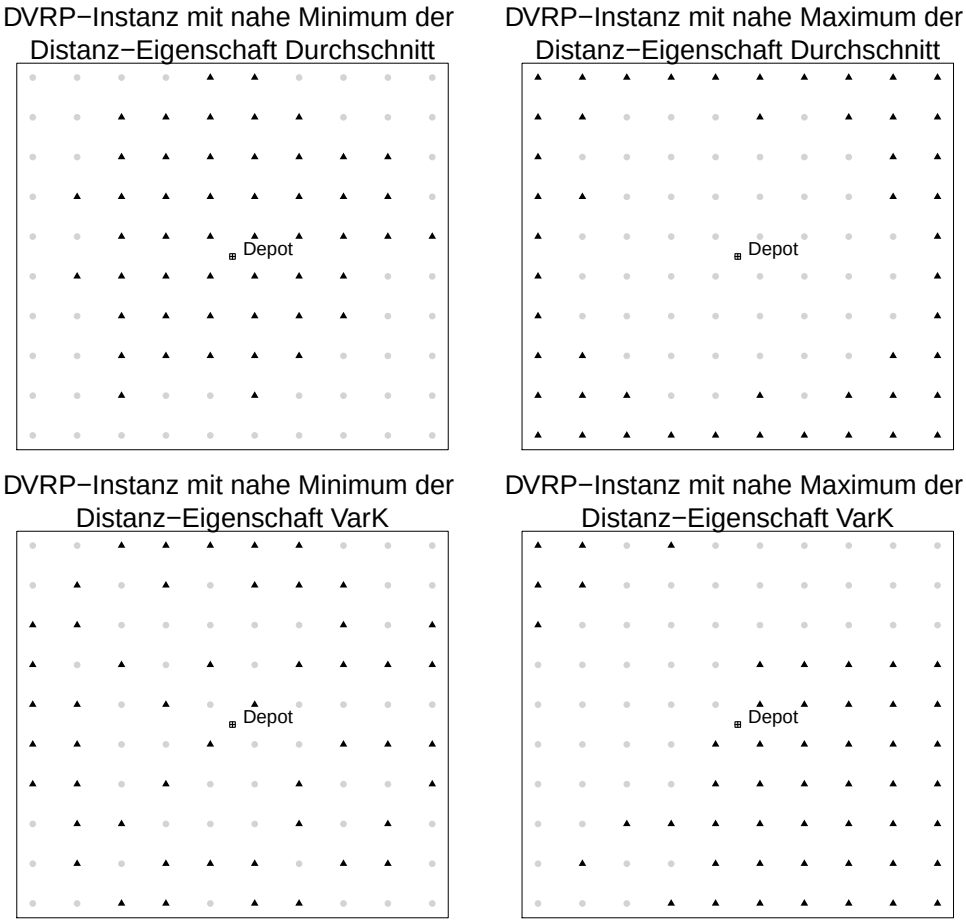

Abbildung 4.10: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Distanz-Eigenschaften Durchschnitt und Variationskoeffizient

Nächster Nachbar-Eigenschaften NNE_{dyn}

Die Nächsten-Nachbar (NN)-Eigenschaften NNE_{dyn} umfassen den Durchschnitt und den Variationskoeffizient der Distanzen $c_{i,k}$ zwischen allen dynamischen Anfragen v_i und der jeweils nächst gelegenen dynamischen Anfrage v_k (vgl. Abbildung 4.11).

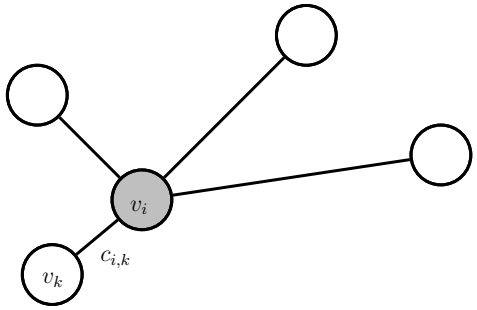

Abbildung 4.11: Dynamische Kundenanfrage v_i mit Distanz $c_{i,k}$ zum nächsten Nachbarn v_k

Die Ergebnisse der Analyse der Wertebereiche der NN-Eigenschaften ist in Abbildung 4.12 dargestellt. Gut ist zu erkennen, dass die durchschnittlichen Distanzen zu den nächsten Nachbarn mit der Größe der Fläche auf der die dynamischen Kundenanfragen verteilt sind korrelieren.

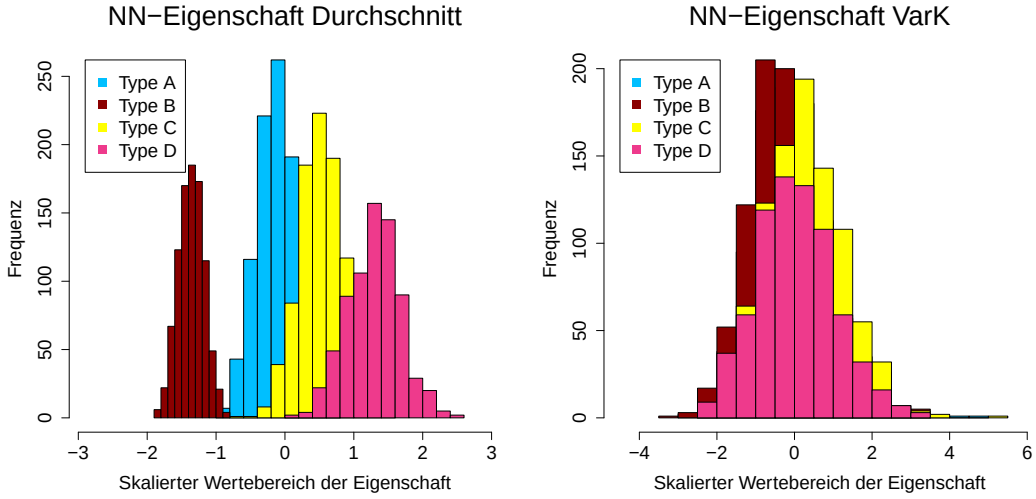

Abbildung 4.12: Wertebereiche der Nächster-Nachbar (NN)-Eigenschaften Durchschnitt und Variationskoeffizient $VarK$ nach Typen von Probleminstanzen (vgl. Abbildung 4.2)

Typ B, bei der alle dynamischen Anfragen im Zentrum der Probleminstanz liegen, weist den kleinsten Mittelwert der Normalverteilung für die NN-Eigenschaft Durchschnitt auf. Instanzen vom Typ A, bei der dynamische Anfragen nur genau auf einer Hälfte der Probleminstanz platziert sind und damit auf einer geringfügig größeren Fläche verteilt sind, weisen einen leicht größeren Mittelwert auf. In Probleminstanzen vom Typ C und D sind die dynamischen Anfragen auf einer größeren Fläche verteilt. Die

NN-Eigenschaften Durchschnitt zeigen größere Mittelwerte der Normalverteilung für diese Typen. Allgemein ist zu erkennen, dass die Wertebereiche der Instanzen von Typ B, C und D teilweise überlagert sind. Die NN-Eigenschaft Durchschnitt gibt einen guten Überblick über die Fläche auf der die dynamischen Anfragen verteilt sind und über die Dichte der Verteilung. Befinden sich viele dynamische Anfragen auf einer kleinen Fläche ist die NN-Eigenschaft Durchschnitt klein im Vergleich zu einer Probleminstanz mit wenigen Anfragen verteilt auf eine große Fläche.

Für die Variationskoeffizient der Distanzen zu den nächsten Nachbarn der dynamischen Kundenanfrage zeigen die Werte für Instanzen vom Typ A, B, C und D fast identische Verteilungen und erlauben so keine Unterscheidung der Probleminstanzen.

Abbildung 4.13 zeigt die DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der NN-Eigenschaften Durchschnitt und Variationskoeffizient. Bei Instanzen mit großen Werten für den durchschnittlichen Abstand zwischen Nachbarn sind die dynamischen Kundeanfragen erwartungsgemäß gleichmäßig, aber mit großem Abstand zueinander auf der Problemfläche verteilt. Bei kleinen Werten für den durchschnittlichen Abstand zwischen Nachbarn zeigt sich nicht wie erwartet eine Bündelung von dynamischen Anfragen auf nur einem Teil der Problemfläche. Die Kundenanfragen sind fast gleichmäßig, nah beieinander, aber noch über die gesamte Fläche verteilt. Eine mögliche Ursache kann die gewählte Anzahl von Generationen des GA sein. Die Unterscheidung der Probleminstanzen für den Variationskoeffizient gestalten sich ebenfalls schwierig. Bei einem hohen Wert für den Variationskoeffizient sind die dynamischen Anfragen eher unregelmäßig auf der Problemfläche verteilt. Es bilden sich größere Gruppen mit einzelnen Ausreißern. Bei einem kleinen Wert sind die dynamischen Anfragen in kleineren Gruppen gleichmäßig über die Fläche der Instanz verteilt.

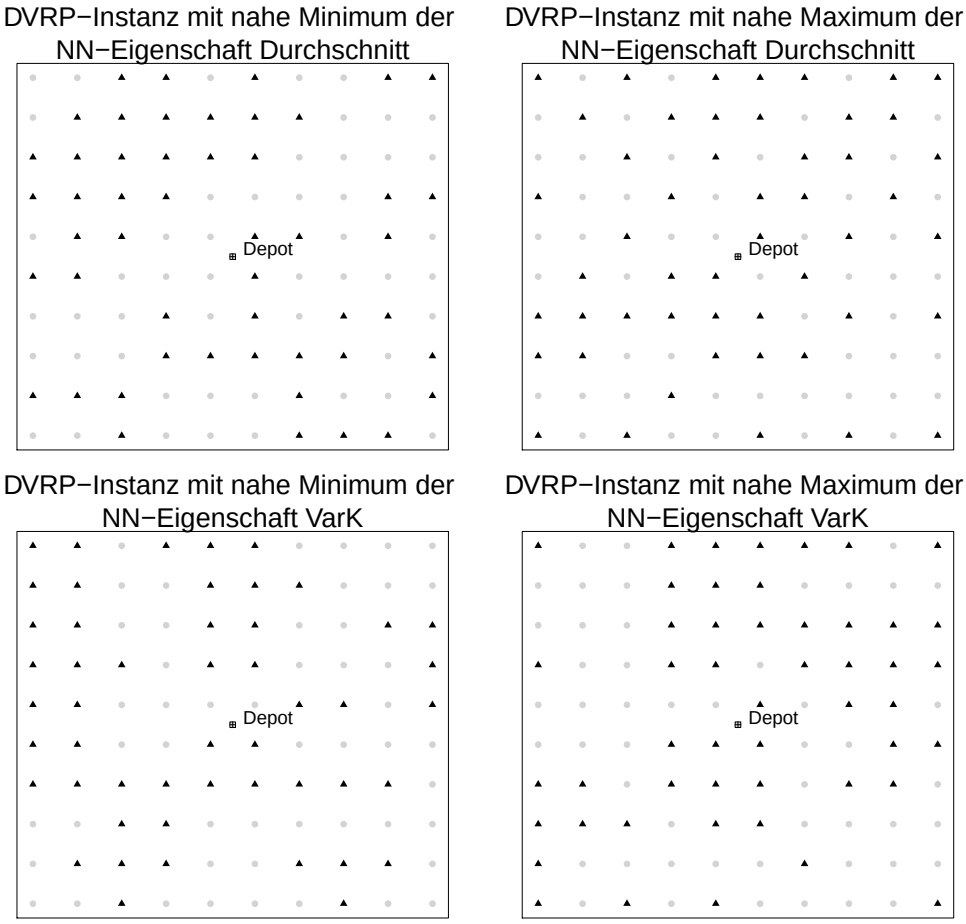

Abbildung 4.13: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Nächster-Nachbar (NN)-Eigenschaften Durchschnitt und Variationskoeffizient

Minimaler Spannbaum-Eigenschaften MST_{dyn}

Die Minimaler Spannbaum (MST)-Eigenschaften MST_{dyn} umfassen den Durchschnitt und den Variationskoeffizient der Distanzen zwischen den Knoten des MST. Ein Spannbaum einer DVRP-Probleminstanz enthält alle dynamischen Kundenanfragen. Die Anfragen werden zusammenhängend über die Kanten verbunden und sind so gewählt, dass keine geschlossenen Pfade (Zyklen) entstehen. Der minimale Spannbaum einer DVRP-Instanz ist beispielhaft in Abbildung 4.14 illustriert. Die Tiefe des Baums entspricht der Summe der Kantenkosten $c_{m,k}$, $c_{i,m}$, $c_{i,l}$ und $c_{l,j}$. Der minimale Spannbaum kann, zum Beispiel, mithilfe des Algorithmus von Kruskal, eingeführt in Kruskal (1956), berechnet werden. Eine mögliche Umsetzung des Algorithmus wird in Abschnitt A.1 erläutert.

Die Ergebnisse der Analyse der MST-Eigenschaften sind in den Histogrammen in Abbildung 4.15 visualisiert. Wie erwartet ist die durchschnittliche Tiefe des minimalen Spannbaums ein guter Indikator für die Größe der Fläche, auf der die dynamischen Kundenanfragen verteilt sind. Die dynamischen Anfragen bei Probleminstanzen vom Typ B sind auf der kleinsten Fläche im Vergleich zu den anderen Typen verteilt und

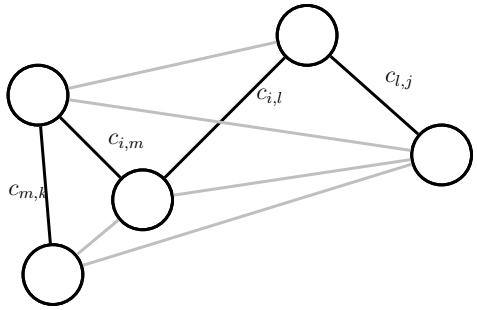

Abbildung 4.14: Der MST (schwarze Kanten) mit Distanzen $c_{m,k}$, $c_{i,m}$, $c_{i,l}$ und $c_{l,j}$ (Kantenkosten)

weisen so auch den kleinsten Mittelwert der durchschnittlichen Distanz des minimalen Spannbaums auf. Dynamische Anfragen von Instanzen vom Typ C sind auf der gesamten Problemfläche verteilt, der Mittelwert der durchschnittlichen Distanz des MST ist hier am größten.

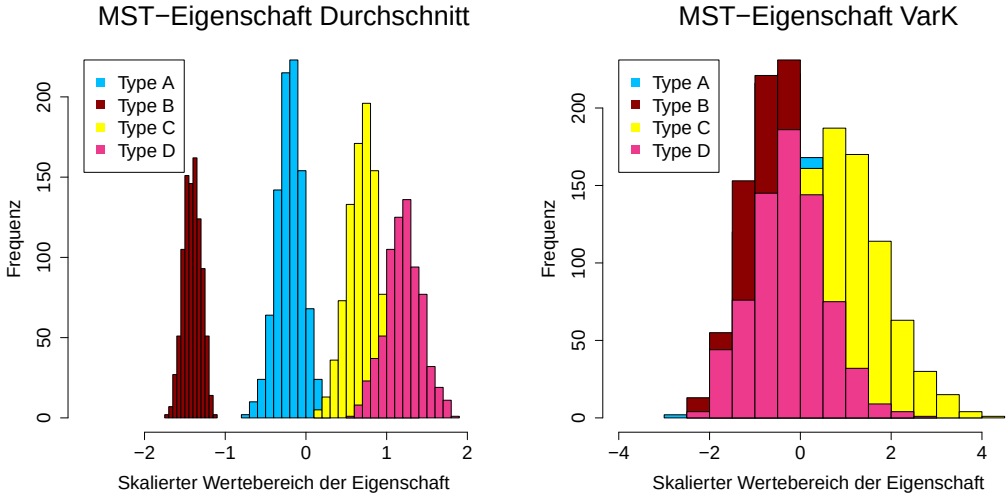

Abbildung 4.15: Wertebereiche der MST-Eigenschaften Durchschnitt und Variationskoeffizient $VarK$ nach Typen von Probleminstanzen (vgl. Abbildung 4.2.)

Der Mittelwert der durchschnittlichen Distanz des minimalen Spannbaums korreliert mit der Fläche auf denen die dynamischen Anfragen verteilt sind. Die Wertebereiche für Typ C und D überschneiden sich teilweise und machen so eine klare Unterscheidung mithilfe der durchschnittlichen Distanz des MST nicht eindeutig. Instanzen der Typen A und B lassen sich hingegen klar voneinander unterscheiden. Auch können die Typen A und B klar von den Typen C und D separiert werden. Für die Variationskoeffizient der Distanzen des minimalen Spannbaums der dynamischen Kundenanfrage zeigen die Werte für Instanzen vom Typ A, B, C und D ähnliche Verteilungen (vgl. Abbildung 4.15) und erlauben so keine Unterscheidung der Probleminstanzen.

Probleminstanzen mit den nahe minimalen und maximalen Ausprägungen für die MST-Eigenschaften sind in Abbildung 4.19 abgebildet. Es zeigt sich, dass die durch-

schnittliche Distanz des MST große Werte annimmt, wenn die Anfragen gleichmäßig, aber maximal voneinander entfernt auf der Problemfläche verteilt sind. An allen Ecken der Fläche befinden sich dynamische Anfragen. Nahezu minimal wird die durchschnittliche Distanz im Spannbaum, wenn die Anfragen eher nah beieinander verteilt sind. Der Variationskoeffizient nimmt große Werte an, wenn die Distanzen zwischen den Kundenanfragen stark variieren. So entstehen große Verbände von Anfragen mit vereinzelten Ausreißern. Ein kleiner Wert für den Variationskoeffizient wird durch eher gleichmäßig verteilte dynamische Anfragen auf der Problemfläche erreicht.

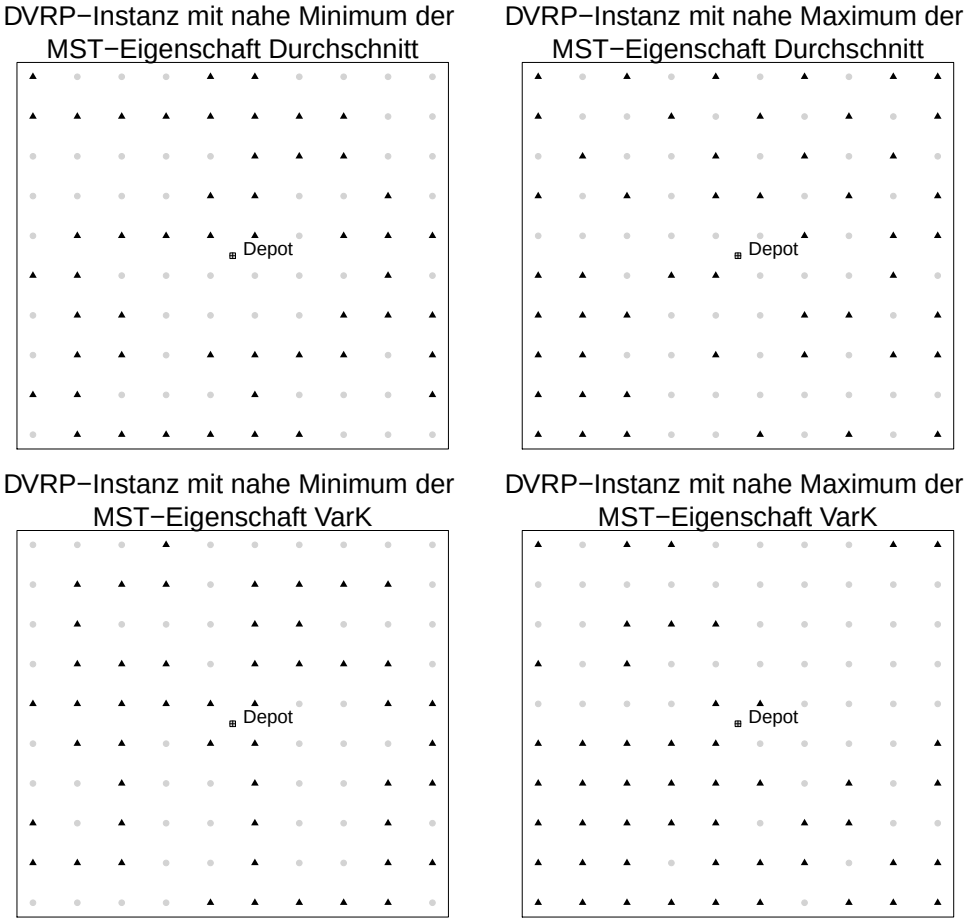

Abbildung 4.16: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der MST-Eigenschaften Durchschnitt und Variationskoeffizient

Cluster-Eigenschaften CE_{dyn}

Die Cluster-Eigenschaften CE_{dyn} umfassen zum einen die Anzahl der Cluster und zum anderen den Durchschnitt aller Abstände $c_{i,cm}$ der dynamischen Kundenanfragen v_i zum zugehörigen Clustermittelpunkt CM (vgl. Abbildung 4.17).

Die Cluster-Eigenschaften werden, wie alle bereits vorgestellten Eigenschaften, mithilfe des R-Pakets tspmeta, eingeführt von Mersmann et al. (2013) berechnet. tspmeta

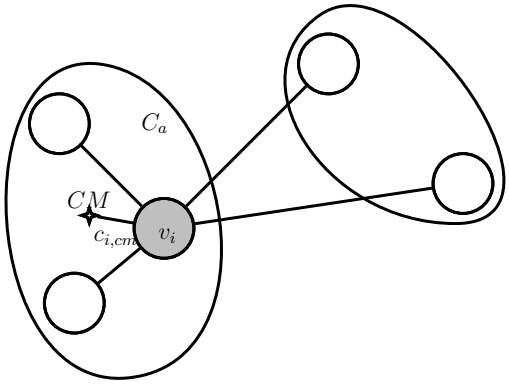

Abbildung 4.17: Distanz $c_{i,cm}$ der dynamischen Kundenanfrage v_i zum Mittelpunkt CM des zugehörigen Clusters C_a

nutzt den *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) Algorithmus, eingeführt in Ester et al. (1996), für das Clustering der Probleminstanzen. DBSCAN basiert auf der Idee von dichteverbundenen Objekten. Zwei Objekte sind dichteverbunden, wenn es eine Reihe von dichten Objekten gibt, die diese beiden miteinander verbinden. Dichte Objekte oder auch Kernobjekte haben mehr als $minPts$ Nachbarn. Ein Cluster bildet sich aus allen Objekten, die mit dichten Objekten eine Nachbarschaftsbeziehung eingehen. Kann ein Objekt keinem Cluster zugeordnet werden bezeichnet man es als Rauschen. Als Eingangsparameter erwartet der Algorithmus die Nachbarschaftslänge ε. Zwei Objekte haben eine Nachbarschaftsbeziehung, wenn die Distanz zwischen beiden nicht größer als ε ist. Der zweite Eingabeparameter $minPts$ definiert das Kernobjekt. Für die Berechnung der Cluster-Eigenschaften gilt $minPts = 5$ und ε entspricht dem 5%-Quantil der Distanzen zwischen dynamischen Kundenanfragen.

Die Ergebnisse der Analyse der Cluster-Eigenschaften, dargestellt in Abbildung 4.18, entsprechen den Erwartungen. Die Wertebereiche der Cluster-Eigenschaften der zu unterscheidenden Typen von Probleminstanzen überlagern sich stark. Kein Typ zeichnet sich durch eindeutig voneinander trennbare Cluster von dynamischen Kundenanfragen aus. Die in Abschnitt 4.1 vorgestellten Bildungsvorschriften verteilen dynamische Kundenanfragen auf einer definierten Fläche gleich. Aus diesem Grund ist eine Unterscheidung der verschiedenen Typen von Instanzen auf Grundlage der Cluster-Eigenschaften nicht möglich.

Abbildung 4.19 zeigt die DVRP-Instanzen mit nahe minimalen und maximalen Ausprägungen der Werte für die Cluster-Eigenschaften Durchschnitt und Anzahl. Die durchschnittliche Distanz der Anfragen zum Clustermittelpunkt ist bei kleinen Clustern kleiner als bei großen. Die durch den GA generierten Instanzen für möglichst kleine bzw. große Ausprägungen für die durchschnittliche Distanz bestätigen diese Annahme. Um die Anzahl der Cluster zu maximieren, erzeugt der Algorithmus eine Probleminstanz mit vielen kleineren Clustern. Interessant ist, dass, um die Anzahl der Cluster zu minimieren,

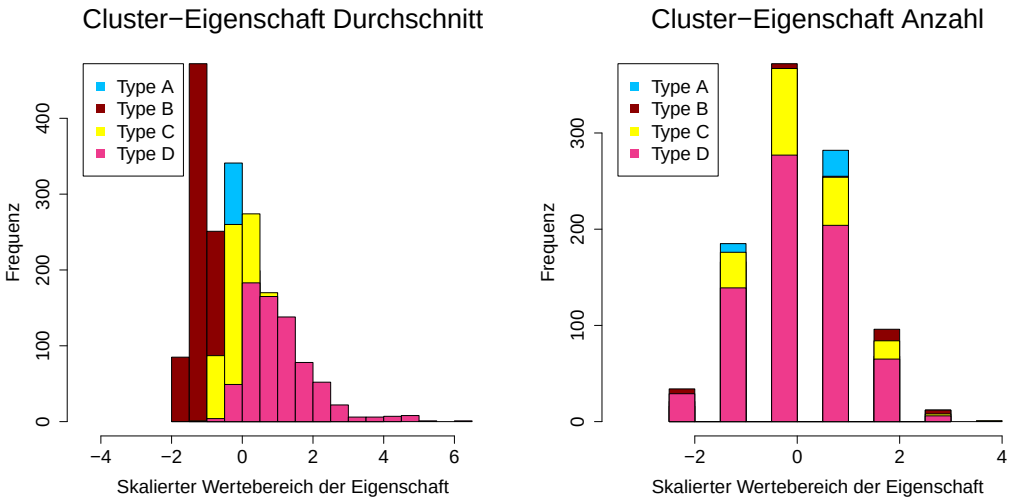

Abbildung 4.18: Wertebereiche der Cluster-Eigenschaften durchschnittliche Distanz der dynamischen Anfrage zum Clustermittelpunkt und Clusteranzahl nach Typen von Probleminstanzen (vgl. Abbildung 4.2.)

der Algorithmus nicht versucht einen großen zusammenhängenden Cluster zu erzeugen, sondern scheinbar sehr kleine Cluster generiert. Hier ist zu vermuten, dass aufgrund der Wahl der Parameter $minPts$ und ε für den Clusteralgorithmus die sehr kleinen Cluster nicht mehr als solche interpretiert und als Rauschen behandelt werden.

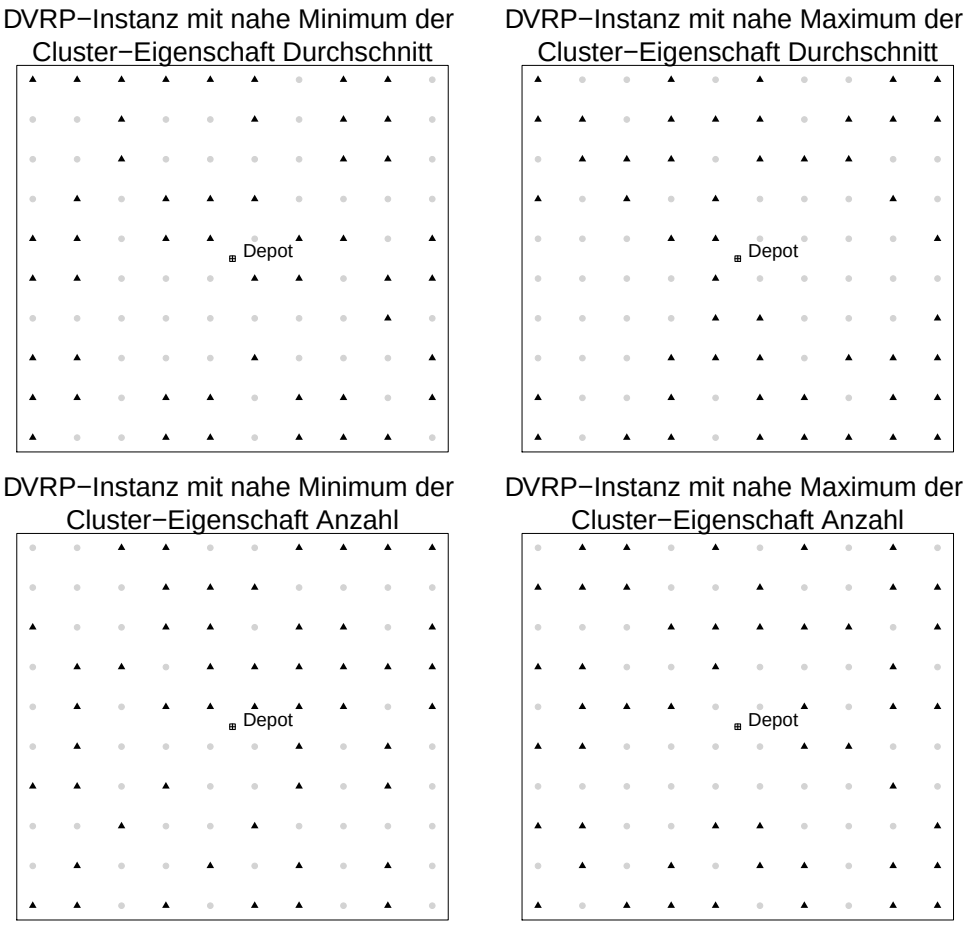

Abbildung 4.19: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Cluster-Eigenschaften Durchschnitt und Anzahl

Zusammenfassung und Diskussion

Verschieden Strukturen der dynamischen Kundenanfragen in einer DVRP-Instanz können mithilfe der Eigenschaften, entwickelt für das TSP, zuverlässig erkannt werden. Tabelle 4.1 fasst die vermutete Aussagekraft der Eigenschaften über die Beschaffenheit der Probleminstanzen zusammen. Häufig lassen die Eigenschaften Rückschlüsse auf die Fläche, auf der die dynamischen Kundenanfragen verteilt sind, vermuten. Gerade große Distanzen zwischen nächsten Nachbarn und im minimalen Spannbaum erfordern die gesamte Fläche der Probleminstanz. Diese Korrelationen zwischen den Eigenschaften visualisiert die Korrelationsmatrix, abgebildet in Abbildung 4.20. Die NN-Eigenschaft Durchschnitt, MST-Eigenschaft Durchschnitt und Distanz-Eigenschaft Durchschnitt korrelieren stark positiv miteinander. Wird der durchschnittliche Abstand einer Kundenanfrage zum zugehörigen Clustermittelpunkt maximiert, wird ebenfalls die gesamte Fläche der Probleminstanz ausgeschöpft, ein Zusammenhang, der sich auch in der Korrelationsmatrix in Abbildung 4.20 wiederfindet. Wird hingegen die Clusteranzahl maximiert entstehen viele kleine Cluster mit Objekten, die einen geringen Abstand zum Clustermittelpunkt haben. Auch diesen Zusammenhang zeigt die Korrelationsmatrix

mit einer negative Korrelation zwischen den Cluster-Eigenschaften Durchschnitt und Anzahl.

Tabelle 4.1: Vermutete Zusammenhänge zwischen der Beschaffenheit der DVRP-Instanz und den Eigenschaften, die die dynamischen Anfragen beschreiben

Eigenschaft	**Vermutete Zusammenhänge**	
	Durchschnitt	**Variationskoeffizient**
Winkel (WE_{dyn})	Verbände bzw. Ketten von dynamischen Anfragen	Verbände bzw. eine Mischung aus Ketten und Verbände von dynamischen Anfragen
Distanz (DE_{dyn})	Fläche, auf der die dynamischen Anfragen verteilt sind	Betrag der Differenz zwischen Höhe und Breite der Fläche, auf der die dynamischen Anfragen verteilt sind; Kleine Gruppen von Anfragen, die weit von einer Hauptgruppe entfernt sind
Nächster Nachbar (NN_{dyn})	Fläche, auf der die dynamischen Anfragen verteilt sind; Dichte der dynamischen Anfragen	Ausreißer
Minimaler Spannbaum (MST_{dyn})	Fläche, auf der die dynamischen Anfragen verteilt sind	Große Verbände mit einzelnen Ausreißern
	Durchschnitt	**Anzahl**
Cluster (CE_{dyn})	Kleine bzw. große Cluster	Anzahl der Cluster; Ausreißer

Zusammenfassend reichen die Eigenschaften dieser Kategorie nicht aus, um die allgemeinen Typen von Instanzen, abgebildet in Abbildung 4.2 zu unterscheiden. Ähnelt sich die Fläche, auf der die dynamischen Anfragen verteilt sind, wie bei den Typen C und D, können diese nicht mehr eindeutig voneinander unterschieden werden. Zu bedenken ist jedoch, dass diese Unterscheidung eventuell im Kontext von Algorithmenselektion gar nicht notwendig ist. Aktuell kann noch keine Aussage darüber getroffen werden, ob Algorithmen bei ähnlich großen Flächen, auf denen die dynamischen Anfragen verteilt sind, wirklich unterschiedliche Lösungsqualitäten aufweisen. Ähnliches gilt für die Winkel- und Cluster-Eigenschaften. Grundsätzlich motiviert die Schwäche der Eigenschaften beim Unterscheiden der Instanztypen das Betrachten von Problemeigenschaften, die auch die

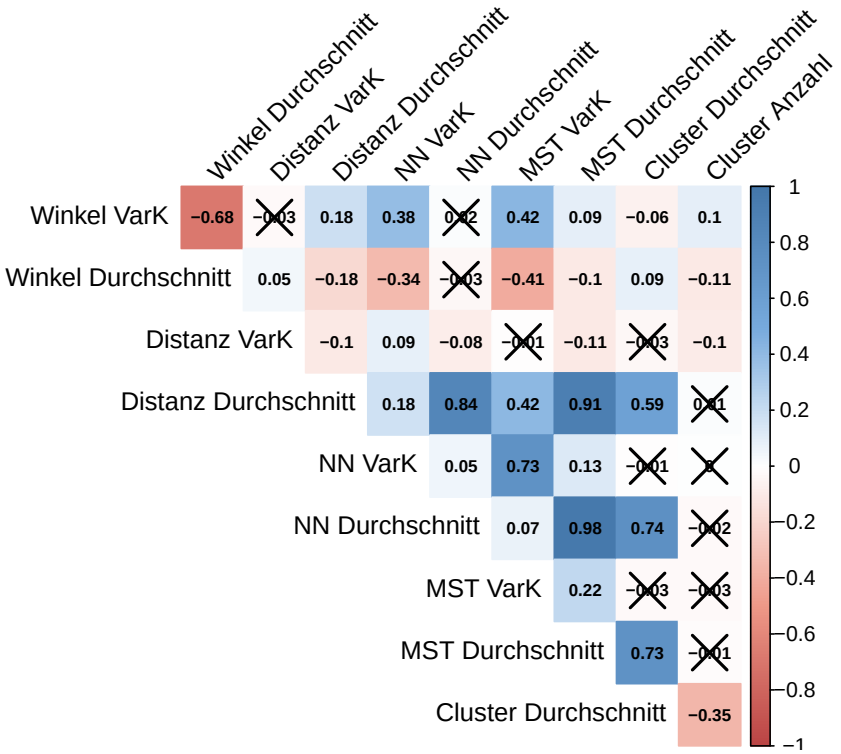

Abbildung 4.20: Korrelationsmatrix der Eigenschaften der dynamischen Anfragen mit Korrelationskoeffizienten (nicht signifikante Beziehungen sind durchgestrichen)

örtlichen Beziehungen zwischen statischen und dynamischen Anfragen berücksichtigen. Eigenschaften dieser Kategorie werden im folgenden Abschnitt vorgestellt.

4.1.3 Eigenschaften der Beziehung zwischen statischen und dynamischen Anfragen

Die im folgenden Abschnitt vorgestellten Eigenschaften für das DVRP berücksichtigen neben den dynamischen auch die statischen Anfragen. Die Eigenschaften versuchen die Beziehungen zwischen den zwei verschiedenen Arten der Kundenanfragen zu erfassen. Diese Beziehungen haben einen großen Einfluss auf die Performance von Algorithmen (Larsen, 2000). Die hier eingeführten Eigenschaften sind vom Autor dieser Arbeit in Auszügen bereits in Mayer et al. (2018*b*) veröffentlicht. Nachfolgend werden die Eigenschaften mit den zugehörigen Untersuchungen vorgestellt.

Winkel-Eigenschaften *WE*

Die Winkel-Eigenschaften *WE* betrachten die Winkelbeziehungen zwischen allen dynamischen Kundenanfragen und den n nächsten statischen Anfragen (Nachbarn). Für die Anzahl der Nachbarn n gilt dabei $n \geq 2$. Ist $n = 2$ wird der Schnittwinkel α, ähnlich wie bei den Winkel-Eigenschaften WE_{dyn} für ausschließlich dynamische Anfragen, zwischen

der dynamischen Anfrage v_i und den drei nächsten statischen Anfragen v_k, v_m und v_l ermittelt (vgl. Abbildung 4.21). Die Berechnungsvorschriften für den Schnittwinkel α ist in Abschnitt A.1 beschrieben.

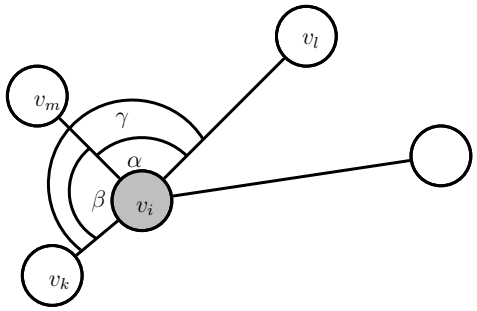

Abbildung 4.21: Schnittwinkel α, β und γ zwischen der dynamischen Kundenanfrage v_i und den drei nächsten statischen Anfragen v_k, v_m und v_l

Betrachtet man die Winkel-Eigenschaften im Kontext von möglichen Lösungen für eine DVRP-Instanz wird ersichtlich, dass voraussichtlich nicht alle dynamischen Kundenanfragen direkt zwischen den nächstgelegenen statischen Anfragen bedient werden können. Aus diesem Grund wird eine erweiterte Nachbarschaft betrachtet. Abbildung 4.21 zeigt die Schnittwinkel α, β und γ zwischen der dynamischen Kundenanfrage v_i und den drei nächsten statischen Anfragen v_k, v_m und v_l. Für die berechneten Winkel für eine dynamische Anfrage v_i für Nachbarschaften N mit $n \geq 2$ Nachbarn werden die Statistiken Maximum, Durchschnitt, Median und Summe berechnet. Für alle berechneten Maxima, Durchschnitte, Mediane und Summen für alle dynamische Anfrage v_i werden die Statistiken Maximum, Durchschnitt, Median, Summe, Standardabweichung, Varianz und Variationskoeffizient betrachtet. In der vorliegenden Arbeit werden die Winkel-Eigenschaften für die Nachbarschaften N mit $n \in \{2; 3; 5\}$ berechnet. Daraus ergeben sich 84 verschiedene Winkel-Eigenschaften für die Charakterisierung einer DVRP-Instanz. Ein Auszug der Analyseergebnisse der Winkel-Eigenschaften, die die Beziehungen zwischen statischen und dynamischen Anfragen beschreiben, ist in Abbildung 4.22 dargestellt.

Gut ist zu sehen, dass sich die Typen B und D mithilfe des Durchschnitts der Schnittwinkeldurchschnitte gut voneinander unterscheiden lassen. Die Mittelwerte der annähernd normalverteilten Werte sind klar voneinander getrennt und die Wertebereiche überschneiden sich kaum. Die Typen B und D können auch gut von den Typen A und C unterschieden werden. Die Typen A und C lassen sich selbst aber nur schwer voneinander abgrenzen. Die Mittelwerte der annähernd normalverteilten Werte unterscheiden sich nur geringfügig und die Wertebereiche überlagern sich stark. Mithilfe der Standardabweichung der Schnittwinkeldurchschnitte lassen sich die Typen A und C geringfügig besser voneinander abgrenzen. Allgemein weisen die Eigenschaften in den verschiedenen Nachbarschaften ähnliche Wertebereiche auf. Alle weiteren der 84 verschiedene Winkel-Eigenschaften zeigen hinsichtlich der Unterscheidung von Instanzen

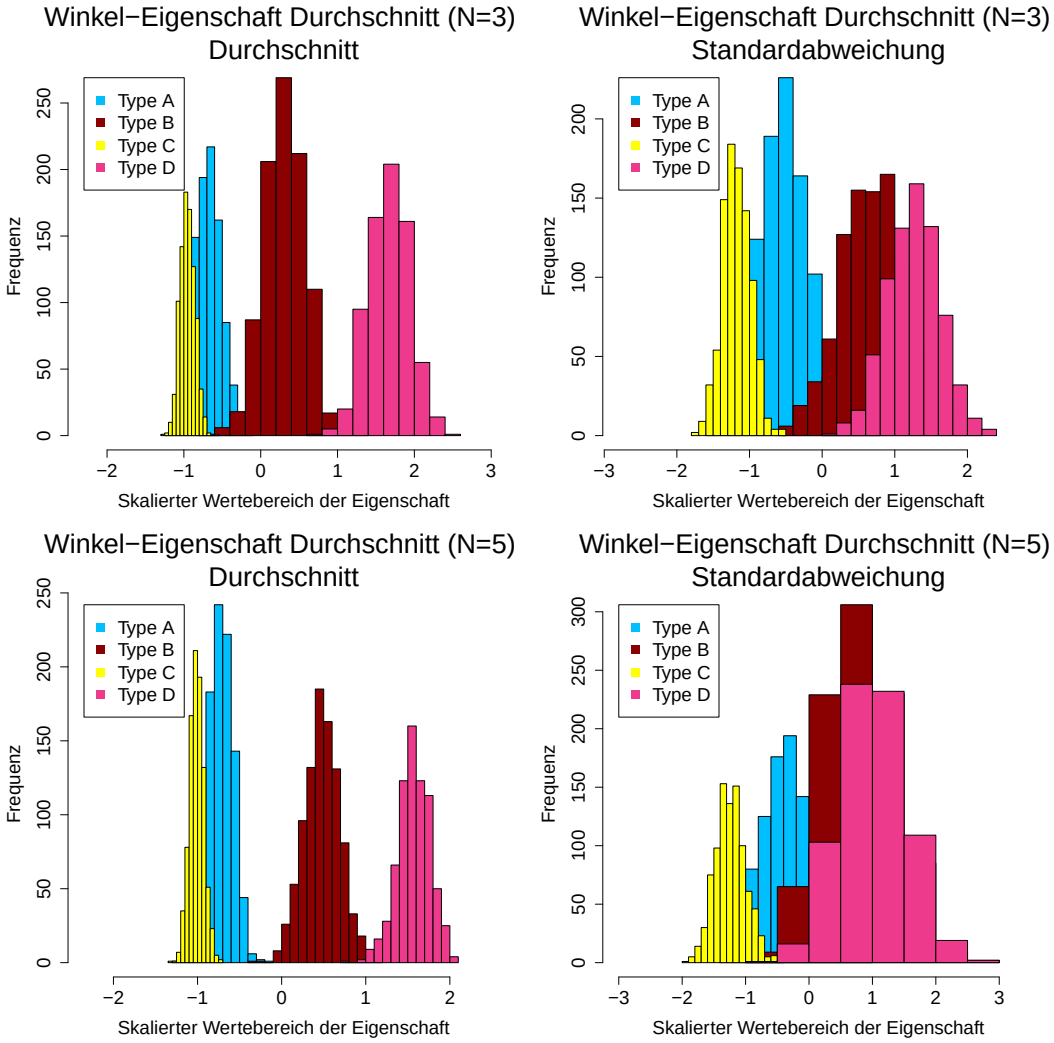

Abbildung 4.22: Wertebereiche der Winkel-Eigenschaften Durchschnitt-Durchschnitt und Durchschnitt-Standardabweichung für die Nachbarschaften $N \in \{3{,}5\}$

der Typen A, B, C und D ähnliches Verhalten. Eine detaillierte Übersicht über die Analyseergebnisse aller Winkel-Eigenschaften ist in Abschnitt A.2 zu finden.

Abbildung 4.23 zeigt die DVRP-Instanzen mit nahe minimalen und maximalen Ausprägungen der Werte für die Winkel-Eigenschaften Durchschnitt-Durchschnitt und Durchschnitt-Standardabweichung (SD) für die Nachbarschaft N mit $n = 3$ (vereinfacht $N = 3$). Für Probleminstanzen mit kleinen Werten für den durchschnittlichen Schnittwinkel zu den nächsten drei Nachbarn sammeln sich die dynamischen Anfragen eher am Rand der Probleminstanz. Für große Winkel verteilen sich die dynamischen Anfragen gleichmäßig auf der Problemfläche. Die Minimierung der Standardabweichung für die durchschnittlichen Schnittwinkel erzeugen auch eher gleichmäßig verteilte dynamische Kundenanfragen. Große Werte für die Standardabweichung führen zu Häufungen und zu gleichmäßig verteilten, einzelnen dynamischen Anfragen.

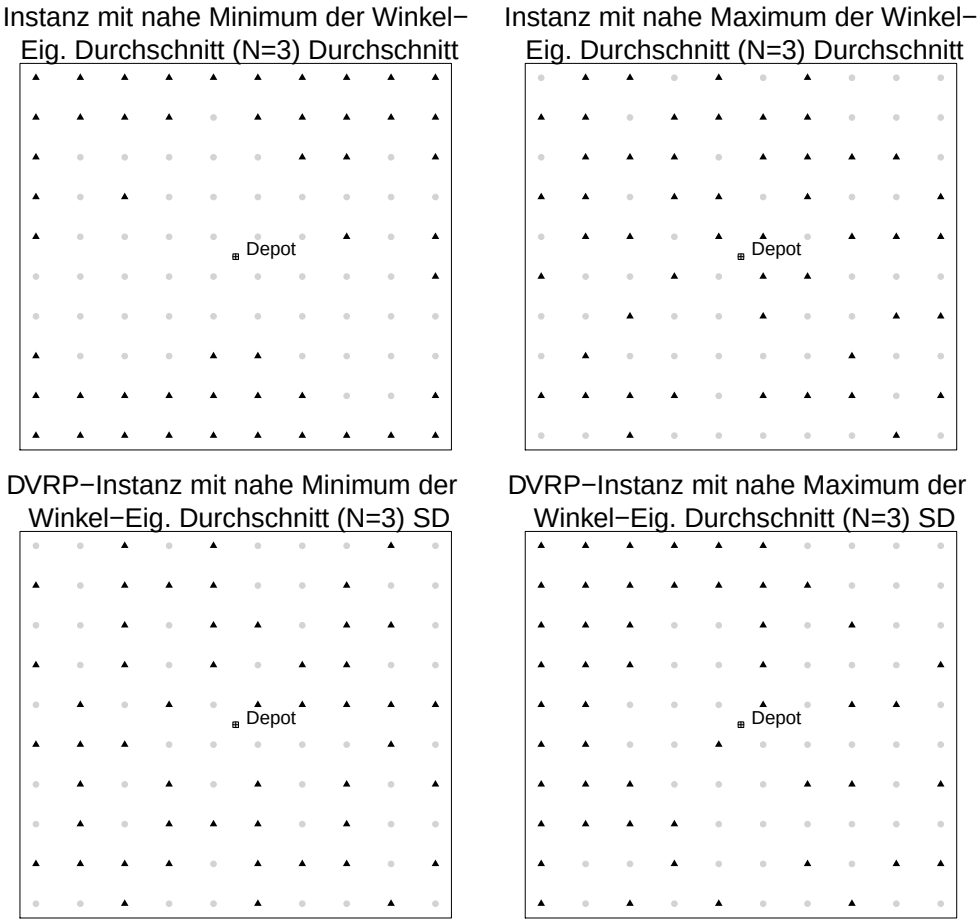

Abbildung 4.23: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Winkel-Eigenschaften Durchschnitt-Durchschnitt und Durchschnitt-Standardabweichung (SD) für die Nachbarschaft N mit $n = 3$

Nächste-Nachbar-Eigenschaften *NNE*

Die Nächsten-Nachbar (NN)-Eigenschaften *NNE* für statische und dynamische Kundenanfragen beschreiben die Distanzen der dynamischen zu den n nächst gelegenen statischen Anfragen. Es werden die Nachbarschaften N mit $n \in \{1; 2; 5\}$ betrachtet. Abbildung 4.24 zeigt die Distanzen $c_{i,m}$ und $c_{i,k}$ zwischen der dynamischen Anfrage v_i und den statischen Anfragen v_m und v_k. Für Nachbarschaften mit $n > 2$ Nachbarn werden die Statistiken Maximum, Durchschnitt, Median und Summe der Distanzen für jede dynamische Kundenanfrage berechnet. Für jede Statistik wird das Maximum, der Durchschnitt, der Median, die Standardabweichung, die Summe, die Varianz und der Varianzkoeffizient als NN-Eigenschaft berechnet. Insgesamt werden so 84 NN-Eigenschaften pro Probleminstanz betrachtet.

Abbildung 4.25 zeigt einen Ausschnitt aus den Analyseergebnissen der NN-Eigenschaften für die in Abbildung 4.2 visualisierten Typen von Probleminstanzen. Gut ist zu sehen, dass sich für die Eigenschaften Durchschnitt-Durchschnitt und Durchschnitt-Standardabweichung sowohl für die Nachbarschaften N mit $n = 2$ und $n = 5$ Nachbarn

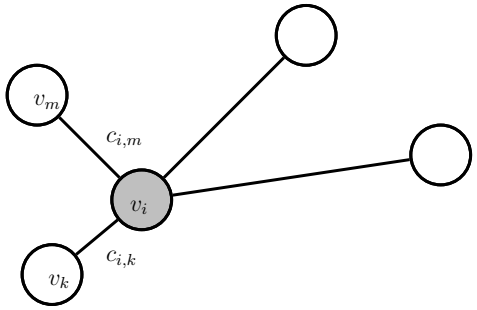

Abbildung 4.24: Distanzen $c_{i,k}$ und $c_{i,m}$ zwischen der dynamischen Kundenanfrage v_i und den zwei nächsten statischen Anfragen v_k und v_m

die Typen A und D klar voneinander abgrenzen lassen. Die Distanzen zwischen dynamischen und statischen Anfragen sind hier signifikant unterschiedlich. Typ D verteilt statische und dynamische Anfragen auf der gesamten Problemfläche gleich, hier sind die Distanzen zwischen den Anfragen am geringsten. Bei Instanzen vom Typ A sind die Anfragen linear voneinander getrennt, weisen somit den größten Abstand untereinander auf. Für Instanzen vom Typ B und C sind die durchschnittlichen Entfernungen zwischen dynamischen und statischen Anfragen ähnlich. Es scheint wenig Einfluss auf die durchschnittliche Distanz zu haben, ob dynamische statische Anfragen umschließen oder umgekehrt. Weitere Ergebnisse der Analyse der Nächsten-Nachbar-Eigenschaften sind in Abschnitt A.2 zu finden.

Abbildung 4.26 zeigt beispielhaft Probleminstanzen mit minimaler und maximaler Ausprägungen der NN-Eigenschaften Durchschnitt-Durchschnitt und Durchschnitt-Standardabweichung (SD) für die Nachbarschaft N mit $n = 2$ Nachbarn. Werden die Distanzen zwischen den dynamischen und statischen Anfragen maximiert bilden sich Verbünde von dynamischen Anfragen am Rand der Probleminstanz. So werden für viele dynamische Anfragen die Abstände zu den nächstgelegenen statischen Anfragen maximal. Wird der Abstand zu den nächsten Nachbarn hingegen minimiert verteilen sich die dynamischen Anfragen gleichmäßig auf der Problemfläche. So sind statische stets in der Nähe von dynamischen Anfragen. Ähnliches Verhalten zeigt sich für die Standardabweichung der Distanzen zu den nächsten Nachbarn. Sind die Anfragen gleichmäßig verteilt, sind die Abstände für alle dynamischen Anfragen zu den statischen nächsten Nachbarn ähnlich. Entstehen hingehen größere Verbünde mit vereinzelten Ausreißern, schwanken die Distanzen stark und haben eine große Standardabweichung.

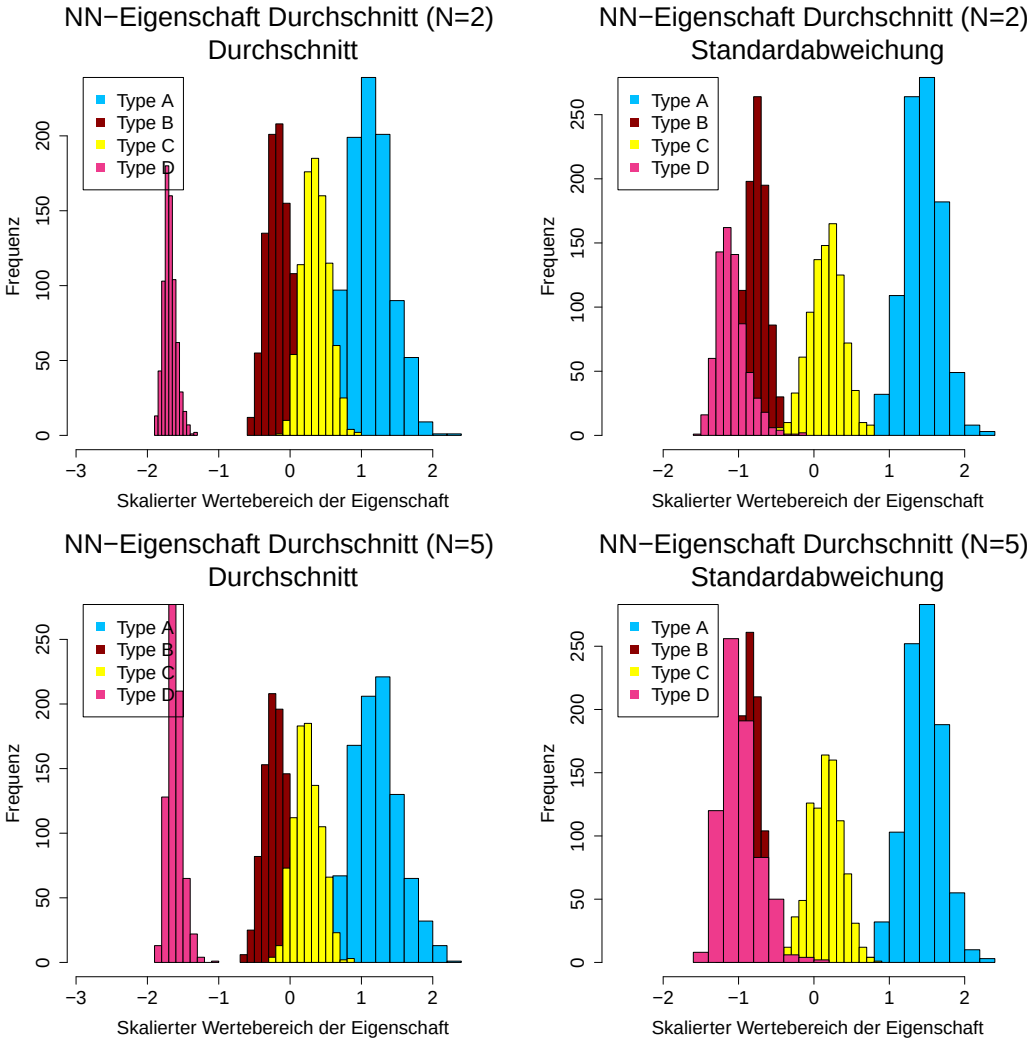

Abbildung 4.25: Wertebereiche der Nächster Nachbar (NN)-Eigenschaften Durchschnitt-Durchschnitt und Durchschnitt-Standardabweichung für die Nachbarschaften N mit $n \in \{2; 5\}$

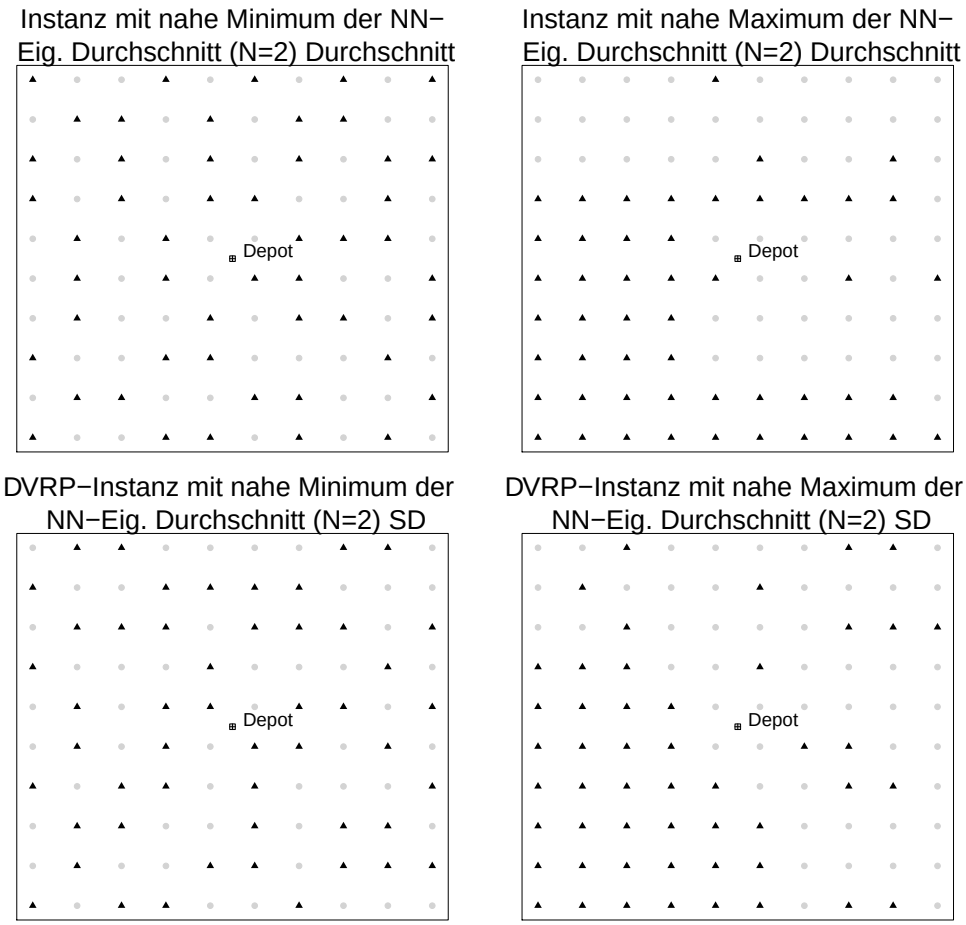

Abbildung 4.26: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der NN-Eigenschaften Durchschnitt-Durchschnitt und Durchschnitt-Standardabweichung (SD) für die Nachbarschaft N mit $n = 2$

Schwerpunkt-Eigenschaften *SE*

Die Schwerpunkt-Eigenschaften *SE* basieren auf dem Abstand der dynamischen und statischen Kundenanfragen zum geometrischen Schwerpunkt der Fläche, die durch die konvexe Hülle aller Anfragen aufgespannten wird. Abbildung 4.27 zeigt den Abstand $c_{i,s}$ zwischen der dynamischen Anfrage v_i und dem geometrischen Schwerpunkt S der Fläche F_{gesamt}. F_{gesamt} wird durch die Anfragen v_k, v_m, v_l und v_j bestimmt. Diese bilden die konvexe Hülle der Probleminstanz. Berechnungsvorschriften für die konvexe Hülle und den geometrischen Schwerpunkt sind in Abschnitt A.1 gegeben.

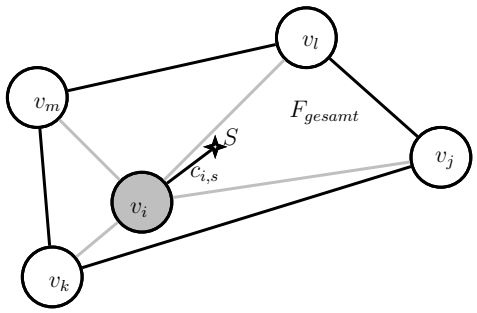

Abbildung 4.27: Distanz $c_{i,s}$ zwischen der dynamischen Kundenanfrage v_i und dem geometrischen Schwerpunkt S der Fläche F_{gesamt}, aufgespannt durch die Kundenanfragen v_k, v_m, v_l und v_j (bilden die konvexe Hülle)

Die Schwerpunkt-Eigenschaft Summe SE_{Summe} ist das Verhältnis zwischen den Summen der Distanzen der dynamischen Kundenanfrage x_{imm} und aller Anfragen x_{tot} zum geometrischen Schwerpunkt (vgl. Gleichung 4.1). Die Berechnungsvorschrift für SE_{Summe} ist in Gleichung 4.2 erfasst. Die Schwerpunkt-Eigenschaft Median SE_{Median} betrachtet das Verhältnis der Mediane der Summen x_{imm} und x_{tot} der Abstände (vgl. Gleichung 4.2).

$$\begin{aligned} x_{imm} &= \sum_{i=1}^{n_{imm}} c_{i,s} \\ x_{tot} &= \sum_{i=1}^{n_{tot}} c_{i,s} \end{aligned} \tag{4.1}$$

$$SE_{Summe} = \frac{x_{imm}}{x_{tot}},\ SE_{Median} = \frac{\widetilde{x_{imm}}}{\widetilde{x_{tot}}} \tag{4.2}$$

Die Eigenschaften SE_{Summe} und SE_{Median} verhalten sich für die erzeugten Probleminstanzen unterschiedlich. Die Visualisierung der Wertebereich der unterschiedlichen Typen von Instanzen, abgebildet in Abbildung 4.28, zeigt das erwartete Verhalten. Das Verhältnis der Summen der Distanzen zum Schwerpunkt ist besonders klein, wenn sich alle dynamischen Anfragen nahe dem Schwerpunkt befinden. Der geometrische Schwerpunkt aller Instanztypen befindet sich in der Nähe des Depots. Das Verhältnis der

Summen wird besonders groß, wenn die dynamischen Anfragen weit vom Schwertpunkt entfernt liegen. Die Instanzen der Typen B und C sind so eindeutig voneinander zu unterscheiden. Die Wertebereiche für die Typen A und B überlagern sich hingegen fast komplett. Die Summe der Abstände der dynamischen Anfragen zum geometrischen Schwerpunkt unterscheiden sich für beide Instanztypen nicht. Für die Mediane der Distanzen zum Schwerpunkt sind die Wertebereiche ähnlich verteilt (vgl. Abbildung 4.28). Der wesentliche Unterschied liegt darin, dass die Mediane der Summen der Distanzen zum Schwerpunkt besonders groß sind, wenn sich alle dynamischen Anfragen nahe dem Schwerpunkt befinden. Besonders klein sind diese, wenn die dynamischen Anfragen weit vom Schwertpunkt entfernt liegen.

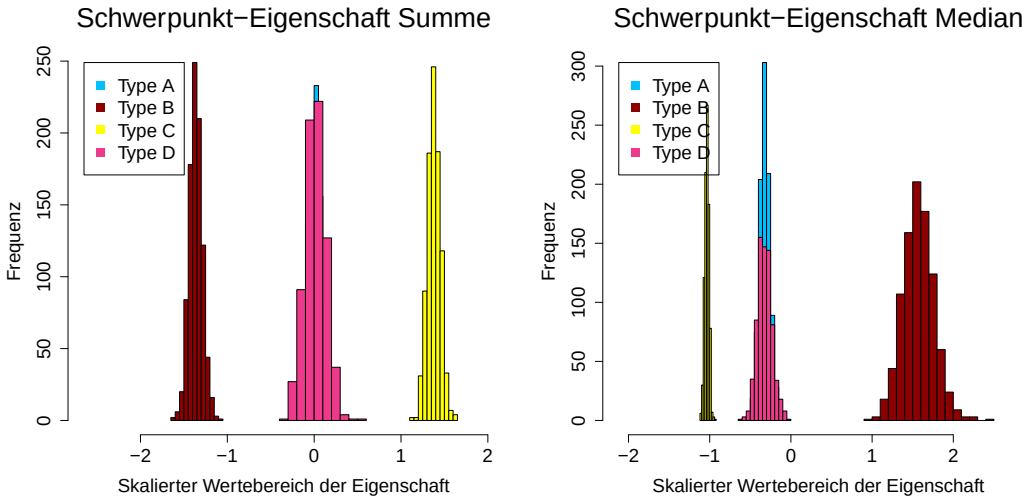

Abbildung 4.28: Wertebereiche der Schwerpunkt-Eigenschaften SE_{Summe} und SE_{Median}

Die Visualisierung der Probleminstanzen mit möglichst minimaler und maximaler Ausprägung der Schwerpunkt-Eigenschaften bestätigt die Analyse der Wertebereiche (vgl. Abbildung 4.29). Ein großer Wert für die Eigenschaft wird erreicht, wenn alle dynamischen Anfragen am Rand der Probleminstanz und weit weg vom geometrischen Schwerpunkt platziert werden. Bündeln sich die dynamischen Anfragen um den Schwerpunkt, wird das Verhältnis der Summen minimiert. Ein gegenteiliger Effekt zeigt sich bei der Schwerpunkt-Eigenschaft Median. Hier weist eine Sammlung der dynamischen Anfragen am Rand der Instanz auf einen kleinen Median hin. Eine Bündelung um den geometrischen Schwerpunkt zeigt sich bei großen Werten von SE_{Median}.

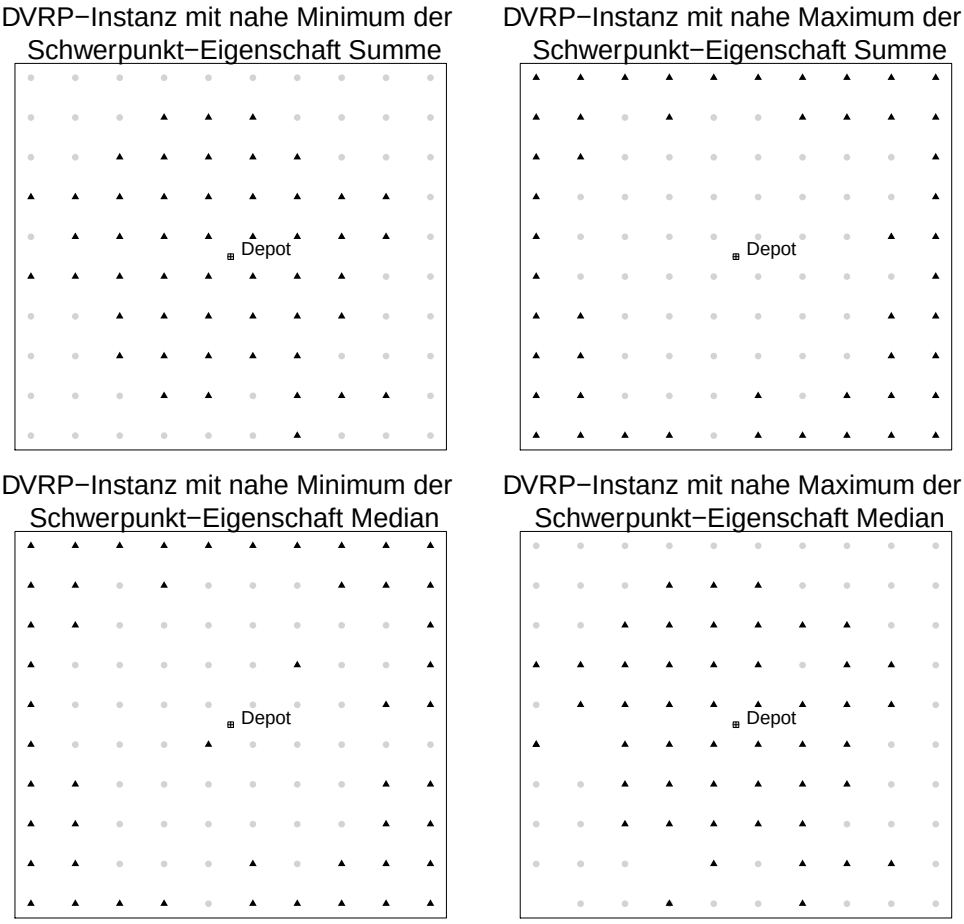

Abbildung 4.29: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Schwerpunkt-Eigenschaft Median und Summe

Distanz-Eigenschaften *LDOD* und *DepotLDOD*

Die Distanz-Eigenschaften basieren auf Distanzen zwischen Anfragen bzw. zwischen Anfragen und dem Depot. Abbildung 4.30 zeigt die dynamische Kundenanfrage v_i mit den zugehörigen Distanzen $c_{i,k}$, $c_{i,m}$, $c_{i,l}$ und $c_{i,j}$ zu allen anderen Kundenanfragen und die statische Anfrage v_j mit den Distanzen $c_{j,k}$, $c_{j,m}$, $c_{j,l}$ und $c_{i,j}$ zu allen anderen Kundenanfragen. Die Distanz-Eigenschaft *Location based Degree of Dynamism* (LDOD), eingeführt vom Autor dieser Arbeit in Mayer et al. (2017), beschreibt die Beziehung zwischen den dynamischen und statischen Anfragen mit dem Verhältnis der Summe der Abstände aller dynamischen Anfragen zu der Summe der Abstände aller Anfragen. Die Berechnungsvorschrift ist in Gleichung 4.3 gegeben. Die Funktion *Distanz* berechnet die euklidische Distanz zwischen zwei Punkten.

$$LDOD = \frac{\sum_{i=1}^{n_{imm}} \sum_{k=1}^{n_{tot}} Distanz(v_i, v_k)}{\sum_{i=1}^{n_{tot}} \sum_{k=1}^{n_{tot}} Distanz(v_i, v_k)} \tag{4.3}$$

Mayer et al. (2017) erbringt zusätzlich den experimentellen Nachweis, dass die Ergebnisse von zwei verschiedenen Algorithmen in Beziehung zum LDOD stehen. Es

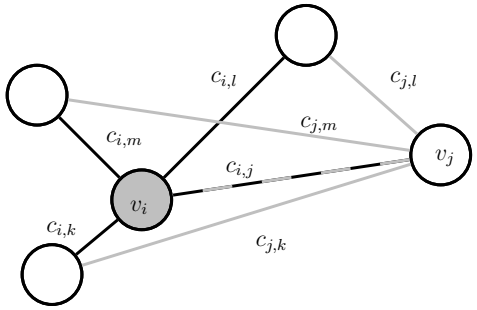

Abbildung 4.30: Distanzen $c_{i,k}$, $c_{i,m}$, $c_{i,l}$ und $c_{i,j}$ der dynamischen Anfrage v_i zu allen anderen Anfragen und Distanzen $c_{j,k}$, $c_{j,m}$, $c_{j,l}$ und $c_{i,j}$ der statischen Anfrage v_j zu allen anderen Anfragen

wird nachgewiesen, dass ein hoher LDOD die Lösungsqualität negativ beeinflusst. Der in dieser Arbeit zusätzlich eingeführte *Depot LDOD* (DepotLDOD) misst nicht den Abstand der Anfragen untereinander, sondern setzt die Abstände der dynamischen Kundenanfragen zum Depot in das Verhältnis zu den Abständen aller Anfragen zum Depot. Die Berechnungsvorschrift für den DepotLDOD ist in Gleichung 4.4 erfasst.

$$DepotLDOD = \frac{\sum_{i=1}^{n_{imm}} Distanz(v_i, v_{Depot})}{\sum_{i=1}^{n_{tot}} Distanz(v_i, v_{Depot})} \tag{4.4}$$

Die Ergebnisse der Analyse der Wertebereiche des LDOD für die verschiedenen Instanztypen, abgebildet in Abbildung 4.31, bestätigten die Ergebnisse aus Mayer et al. (2017). Gut ist zu sehen, dass die Typen B und C sehr gut voneinander getrennt werden können. Hier kommt es zu keiner Überlagerung der Wertebereiche. Die Probleminstanzen des Typs A und D lassen sich nicht mithilfe der Distanz-Eigenschaften voneinander unterscheiden. Hier überlagern sich die Wertebereiche stark. Bei Probleminstanzen mit maximaler Ausprägung des LDOD wird das Verhältnis der Summen der Distanzen zwischen den dynamischen und allen Anfragen maximiert. Die dynamischen Anfragen finden sich aus diesem Grund eher am Rand der Problemfläche. Hier sind die Distanzen zu den innen liegenden Anfragen maximal. Das Verhältnis der Summen der Distanzen zwischen dynamischen und allen Anfragen wird minimiert, wenn sich die dynamischen Anfragen im Zentrum der Probleminstanz bündeln. In allen Typen von Instanzen befindet sich das Depot im Zentrum der Problemstellung (vgl. Unterabschnitt 4.1.1). Aus diesem Grund verhalten sich die Wertebereiche des DepotLDOD ähnlich. Die angestellten Überlegungen bestätigen die Suche nach Probleminstanzen mit minimaler und maximaler Ausprägung der Distanz-Eigenschaft LDOD und DepotLDOD. Die Instanzen mit entsprechenden Ausprägungen der Eigenschaften sind in Abbildung 4.32 abgebildet. Die Probleminstanz mit kleinem Wert für die Eigenschaften ähneln Probleminstanzen vom Typ B (vgl. Abbildung 4.2). Die Probleminstanz mit großem Wert für die Eigenschaften ähneln Probleminstanzen vom Typ C (vgl. Abbildung 4.2).

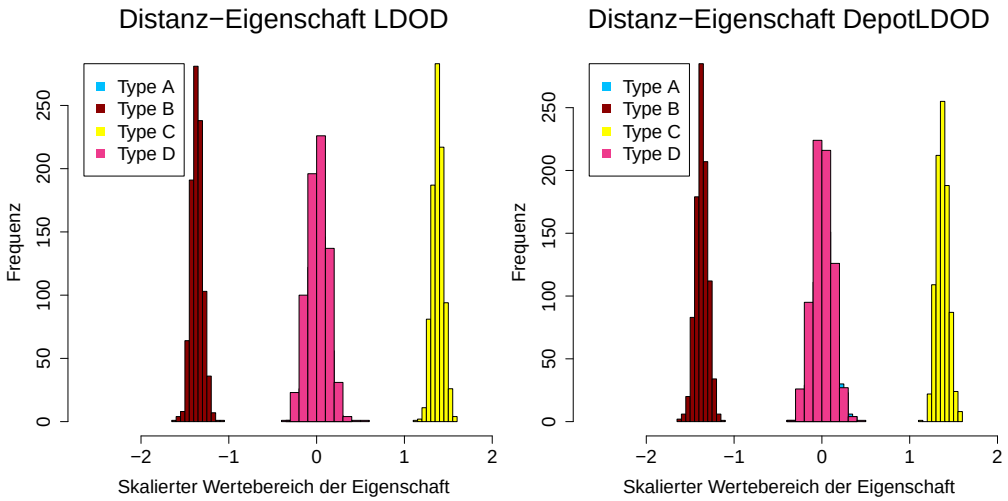

Abbildung 4.31: Wertebereiche der Distanz-Eigenschaften LDOD und DepotLDOD

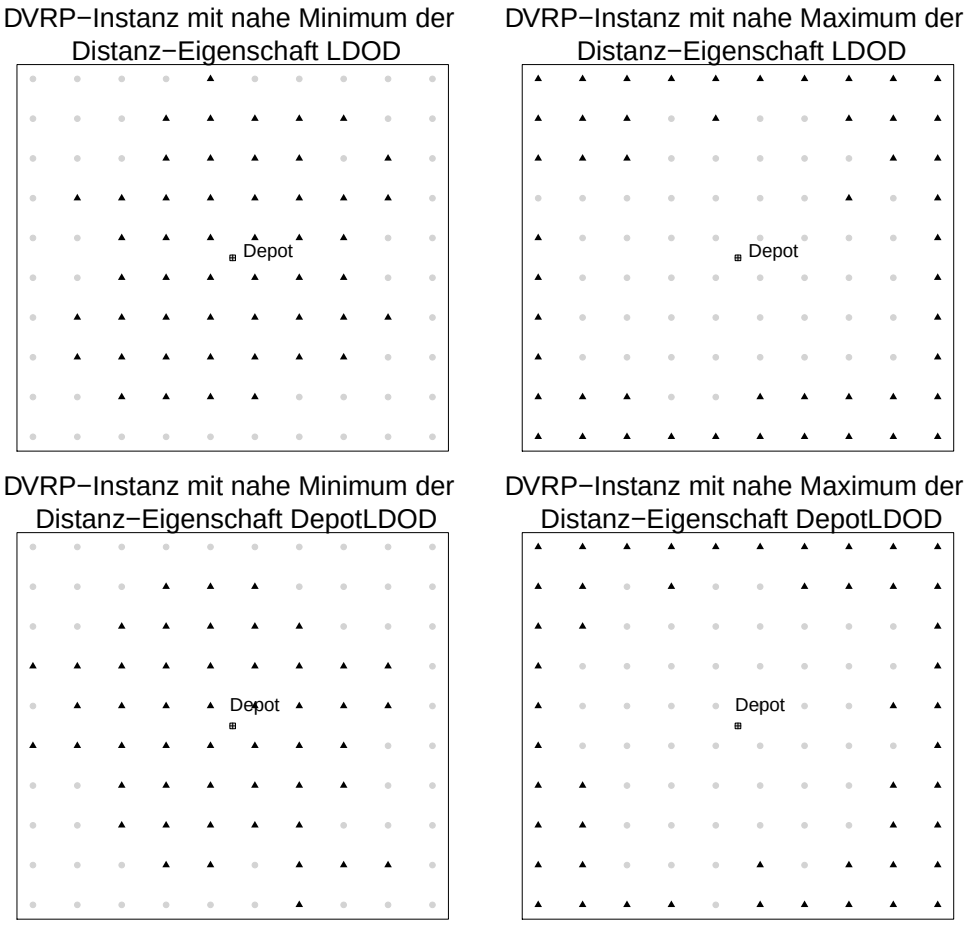

Abbildung 4.32: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Distanz-Eigenschaften LDOD und DepotLDOD

Flächen-Eigenschaft *FE*

Die Flächen-Eigenschaft *FE* beschreibt die Beziehungen zwischen statischen und dynamischen Kundenanfragen mit den Flächen, die die jeweiligen Anfragen aufspannen

(Bounding Box). Abbildung 4.33 zeigt die Fläche F_{alle} die durch alle Kundenanfragen v_k, v_m, v_l und v_j der konvexen Hülle der Probleminstanz aufgespannt wird. Die Anfragen der konvexen Hülle aller dynamischer Anfragen v_l und v_j spannen die Fläche $F_{dynamisch}$. Die Flächen-Eigenschaft FE berechnet sich aus dem Verhältnis von $F_{dynamisch}$ zu F_{alle}. Die Berechnungsvorschrift für die FE ist in Gleichung 4.5 gegeben.

$$FE = \frac{F_{dynamisch}}{F_{alle}} \tag{4.5}$$

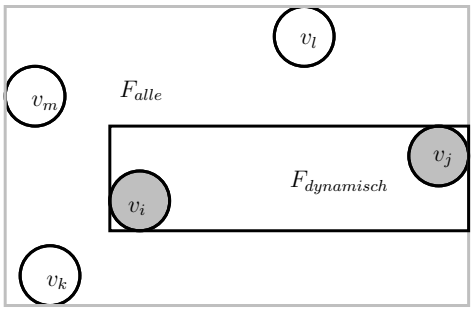

Abbildung 4.33: Maximale Fläche F_{alle}, die alle Anfragen v_k, v_m, v_l und v_j der konvexen Hülle der Probleminstanz aufspannen und maximale Fläche $F_{dynamisch}$, die die konvexe Hülle aller dynamischen Anfragen v_l und v_j aufspannen

Die Analyseergebnisse der Wertebereiche abgebildet in Abbildung 4.34 zeigt das erwartete Verhalten. Für die Instanztypen B und A sind die Flächen, die die dynamischen Anfragen aufspannen, im Verhältnis zur Gesamtfläche klein. Für Instanzen vom Typ C ist das Verhältnis der Flächen immer 1. Die dynamischen Anfragen sind hier am Rand der Probleminstanz, so können keine größeren Flächen durch zusätzliche statische Anfragen aufgespannt werden. Für Probleminstanzen vom Typ D ist das Verhältnis der Flächen oft nahe 1. Hier gilt, dass durch die gleichmäßige Verteilung der statischen und dynamischen Anfragen auf der Problemfläche, die aufgespannten Flächen nur geringfügig voneinander abweichen.

Die Suche nach Probleminstanzen mit minimaler und maximaler Ausprägung der Flächen-Eigenschaft zeigt das erwartete Ergebnis (vgl. Abbildung 4.35). Große Werte (Ausprägungen nahe 1) werden erzielt, wenn mindestens eine dynamische Anfrage an jeder problembegrenzenden Kante platziert ist. Bei kleinen Werten für die Flächen-Eigenschaft sind die dynamischen Anfragen eher zentriert und sind nicht über die gesamte Problemfläche verteilt. Der GA findet hier zwar eine Probleminstanz, bei der die dynamischen Anfragen nicht auf der gesamten Problemfläche verteilt sind, Probleminstanzen mit kleineren Ausprägungen von FE sind aber durchaus denkbar.

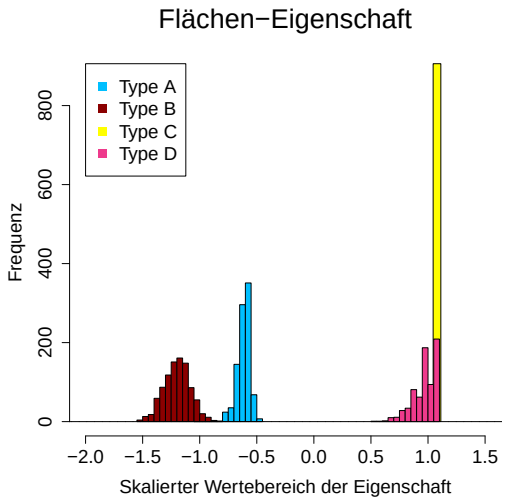

Abbildung 4.34: Wertebereiche der Flächen-Eigenschaft

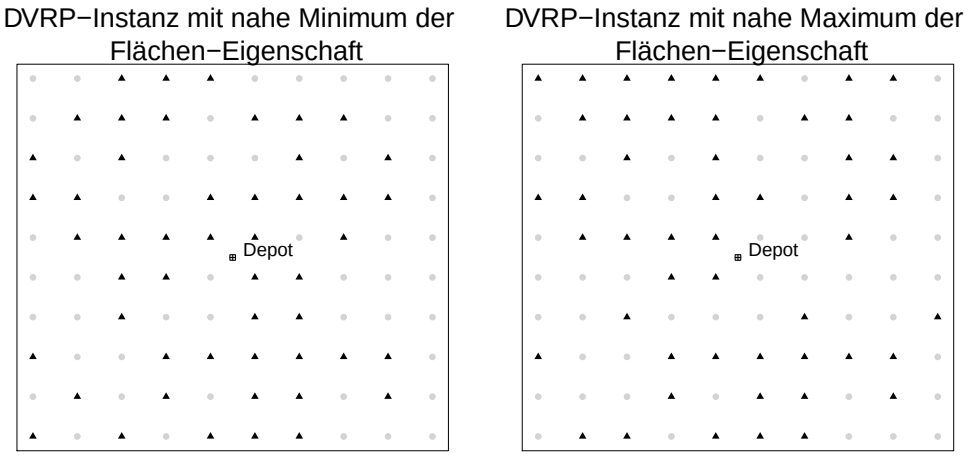

Abbildung 4.35: DVRP-Instanzen mit nahe minimaler und maximaler Ausprägungen der Flächen-Eigenschaft

Zusammenfassung und Diskussion

Die Eigenschaften, die die Beziehung zwischen dynamischen und statischen Anfragen beschreiben, eignen sich gut, um die verschiedenen Typen von Instanzen (vgl. Abbildung 4.2) voneinander zu unterscheiden. So können, zum Beispiel, mit den Winkel-Eigenschaften die Typen A und D klar voneinander abgegrenzt werden. Mithilfe der Flächen-, Schwerpunkt- oder Distanz-Eigenschaften können die Typen B und C unterschieden werden. Darüber hinaus geben die Eigenschaften Auskunft über verschiedene Strukturen in Probleminstanzen, die in Tabelle 4.2 zusammengefasst sind.

Die Probleminstanzen, die durch die Suche nach nahe minimalen bzw. maximalen Ausprägungen der verschiedenen Eigenschaften entstanden sind, weisen teilweise ähnliche Strukturen auf. Die Untersuchung der Eigenschaften auf Korrelationen bestätigt das ähnliche Verhalten von Eigenschaften bei den generierten Typen von Instanzen. So

Tabelle 4.2: Vermutete Zusammenhänge zwischen der Beschaffenheit einer DVRP-Instanz und den Eigenschaften, die die Beziehung zwischen statischen und dynamischen Anfragen beschreiben

Eigenschaft	**Vermutete Zusammenhänge**	
Schwerpunkt (*SE*)	Am Problemrand verteilte dynamische Anfragen oder Verband im Zentrum	
Distanz (*LDOD*, *DepotLDOD*)	Am Problemrand verteilte dynamische Anfragen oder Verband im Zentrum	
Fläche (*FE*)	Verband mit allen dynamischen Anfragen, bzw. dynamische Anfragen, die bis zu den Rändern der Problemfläche verteilt sind	
	Durchschnitt	**Standardabweichung**
Winkel (*WE*)	Gleichmäßig bzw. eher am Rand der Problemfläche verteilte dynamische Anfragen	Verbände am Rand mit einzelnen Ausreißern bzw. eher gleichmäßig verteilte Anfragen
Nächster-Nachbar (*NNE*)	Großer Verband am Rand mit einzelnen Ausreißern bzw. eher gleichmäßig verteilte dynamische Kundenanfragen	Großer Verband am Rand mit einzelnen Ausreißern bzw. eher gleichmäßig verteilte dynamische Kundenanfragen

sind die Flächen-Eigenschaft, die Schwerpunkt-Eigenschaft und der LDOD stark positiv zueinander korreliert (vgl. Abbildung 4.36). Sind die dynamischen Eigenschaften am Rand der Probleminstanz, maximiert sich die Fläche, die durch die Anfragen aufgespannt wird. Zusätzlich wird der Abstand der dynamischen Anfragen zu allen Anfragen und zum geometrischen Schwerpunkt maximal. Die Winkel- und NN-Eigenschaften weisen auch positive Korrelationen zwischen den verschiedenen Nachbarschaften auf. Große Werte werden durch das Hinzufügen weiterer Werte größer. Kleinere Werte bleiben beim Hinzufügen neuer Werte im Verhältnis kleiner. Die Winkel- und Nächste-Nachbar-Eigenschaften sind zueinander stark negativ korreliert.

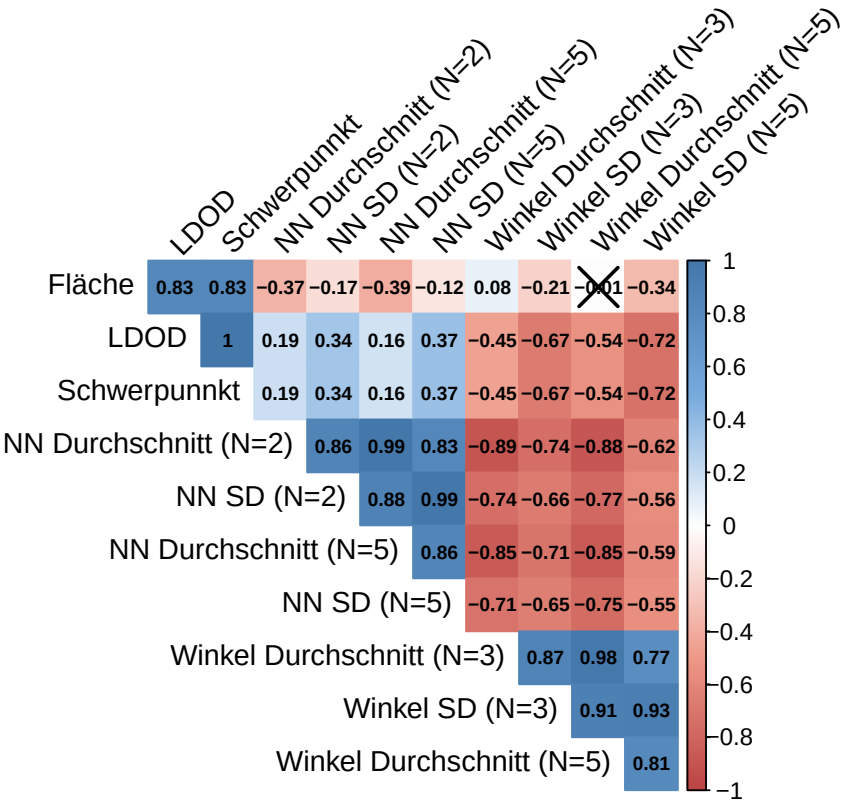

Abbildung 4.36: Korrelationsmatrix der Eigenschaften, die die Beziehung zwischen statischen und dynamischen Kundenanfragen beschreiben mit Korrelationskoeffizienten (nicht signifikante Beziehungen sind durchgestrichen)

4.1.4 Verallgemeinerung und Evaluation

In den folgenden Abschnitten wird kurz eine mögliche Verallgemeinerung der Eigenschaften diskutiert und eine abschließende Evaluation der Problemeigenschaften vorgestellt. Es wird gezeigt, dass die neu eingeführten Eigenschaften eine solide Grundlage bilden, um Probleminstanzen zu charakterisieren. Damit ist der Grundstein gelegt, um Probleminstanzen voneinander zu unterscheiden und so Algorithmen sinnvoll miteinander vergleichen zu können. Die Eigenschaften bilden die Voraussetzung für die automatisierte Auswahl von Algorithmen. Mithilfe der Experimente, durchgeführt im Zuge der automatisierten Algorithmenselektion, kann ergründet werden, welche Problemeigenschaften einen wesentlichen Einfluss auf die Performance von Algorithmen haben. Diese Eigenschaften müssen bei der Erstellung für Prognosemodelle für DVRP berücksichtigt werden.

Verallgemeinerung der Eigenschaften

Nach dem Lemma 4.1.1 sind die Kundenanfragen in jeder möglichen dynamischen Problemstellung mindestens durch einen Ort beschrieben. Unter anderem konzentrieren sich alle betrachteten Eigenschaften aus diesem Grund auf dieses Merkmal. Darüber hinaus können Anfragen in komplexeren Problemstellungen weitere Charakteristika aufweisen (vgl. Unterabschnitt 2.1.1 und Tabelle 2.2). Grundsätzlich ist es möglich auch mit diesen Merkmalen Eigenschaften für eine DVRP-Instanz zu konstruieren. Eine einfache verallgemeinerte Eigenschaft E für ein beliebiges, numerisches Merkmal m einer Kundenanfrage ist in Gleichung 4.6 beschrieben.

$$E = \frac{\sum_{i=1}^{n_{imm}} m_i}{\sum_{i=1}^{n_{tot}} m_i} \tag{4.6}$$

Die Konstruktion der verallgemeinerten Eigenschaften E folgt dabei den DVRP-Eigenschaften DOD (vgl. Unterabschnitt 2.1.3), Schwerpunkt, Fläche und Distanz. Bei all diesen Eigenschaften wird der dynamische Problemanteil am gesamten Problem berechnet. Der zugrunde liegende Gedanke ist dabei, dass sich Algorithmen bei großem dynamischen Problemanteil anders verhalten müssen als bei geringem, um gute Ergebnisse zu erzielen. So wird, zum Beispiel, ein Algorithmus, der viel Platz für den Transport von Gütern, angefragt von dynamischen Anfragen, einplant, weniger effizient Anfragen bedienen können, wenn der dynamische Problemanteil gering ist. Aufgrund der betrachteten Problemstellungen (vgl. Abschnitt 2.3) wird die verallgemeinerte Eigenschaft E in der vorliegenden Arbeit nicht weitergehend betrachtet.

Evaluation der Problemeigenschaften

Mithilfe der visuellen Analyse der Wertebereiche der eingeführten DVRP-Eigenschaften ist der Nachweis erbracht, dass sich die Eigenschaften sehr gut eignen, um die in Abbildung 4.2 abgebildeten Instanztypen voneinander zu unterscheiden. Es soll an dieser Stelle aber zusätzlich untersucht werden, ob sich mithilfe einer einfachen Clusteranalyse die unterschiedlichen Typen der Probleminstanzen klar identifizieren lassen. Für die Clusteranalyse der Problemeigenschaften werden für alle 1.000 erzeugten DVRP-Instanzen die in Unterabschnitt 4.1.2 und Unterabschnitt 4.1.3 eingeführten DVRP-Eigenschaften berechnet. Über alle erzeugten Instanzen hinweg konstante Eigenschaften wie der DOD oder der EDOD finden keine Berücksichtigung. Der entstehende Datensatz wird mithilfe des Clusteralgorithmus K-Means untersucht. Das Ziel dieser Untersuchung ist es, die vier unterschiedlichen Typen der Instanzen im Ergebnis der Analyse als Cluster wiederzufinden. Der K-Means Algorithmus zeichnet sich durch seine Einfachheit und Effizienz aus und wurde aus diesem Grund für die Clusteranalyse ausgewählt. Im Wesentlichen wird durch K-Means eine vorher bekannte Anzahl von Clustern aus einer Menge von ähnlichen Objekten gebildet. Der Algorithmus wird mit einer zufälligen Positionierung der Clustermittelpunkte initialisiert und durchläuft im Anschluss iterativ einen Zuweisungsschritt (assignment step) und einen Anpassungsschritt (update step) (MacKay,

2003). Im Zuweisungsschritt werden die zu clusternden Objekte dem nächstgelegenen Clustermittelpunkte zugeordnet. Im Anpassungsschritt werden die Clustermittelpunkte hin zu den Mittelpunkten der zugeordneten Objekte verschoben. Ändert sich die Zuordnung von Objekt auf Clustermittelpunkte nicht mehr, terminiert der Algorithmus.

Datenstandardisierung ist einer der wichtigsten Schritte in der Datenvorverarbeitungen bei Clusteranalysen (Mohamad und Usman, 2013). Aus diesem Grund werden alle berechneten Eigenschaften für die Probleminstanzen (Datensatz) mithilfe der z-Transformation standardisiert. Bei der z-Transformation werden alle Werte x einer Datenreihe mit dem Mittelwert $\bar{x}$ der Datenreihe subtrahiert und durch die Standardabweichung sd dividiert. Die Berechnungsvorschrift für den transformierten Wert Z aus x ist in Gleichung 4.7 dargestellt.

$$Z = \frac{x - \bar{x}}{sd} \tag{4.7}$$

Obwohl die Anzahl der Cluster gleich der Anzahl der zu unterscheidenden Instanztypen ist (siehe Abbildung 4.2), wird der Datensatz mithilfe der Gap-Statistik-Methode, eingeführt in Tibshirani et al. (2001), untersucht. Die Gap-Statistik-Methode versucht die optimale Anzahl möglicher Cluster in einem Datensatz zu bestimmen. Die Methode clustert den Datensatz K und die Nullreferenz KB (zufällig erzeugter Datensatz mit gleichverteilten Objekten) mit unterschiedlichen Anzahlen von Clustern. Die Nullreferenz KB wird B mal erzeugt. Üblicherweise ist B gleich 10. Für jedes Clusterergebnis wird die Kompaktheit W_K und W^*_{KB} über alle Cluster M mithilfe der Summe der Distanzen der Objekte x in einem Cluster C berechnet. Die Berechnung der Summe D ist in Gleichung 4.8 und die Berechnung der Kompaktheit W in Gleichung 4.9 dargestellt.

$$D = \sum_{x_i \in C} \sum_{x_j \in C} \|x_i - x_j\|^2 \tag{4.8}$$

$$W = \sum_{m=1}^{M} \frac{1}{2n_m} D_m \tag{4.9}$$

Im Anschluss wird der Abstand $Gap(k)$ zwischen den Werten für die Kompaktheit W_K und W^*_{KB} und die Standardabweichung $sd(K)$ nach der Berechnungsvorschrift in Gleichung 4.10 bestimmt. Die Gleichung für die Berechnung der Standardabweichung ist in Anhang A zu finden.

$$Gap(k) = \frac{1}{B} \sum_{b=1}^{B} log(W^*_{KB}) - log(W_K) \tag{4.10}$$

Mithilfe des Simulationsfehlers s_k und dem Wert für Gap kann die optimale Anzahl an Clustern nach Gleichung 4.12 ermittelt werden. Die Berechnungsvorschrift für s_k ist in Gleichung 4.11 erfasst.

$$s_k = sd(K)\sqrt{1 + \frac{1}{B}} \tag{4.11}$$

$$Gap(k) \geq Gap(k+1) - s_{k+1} \tag{4.12}$$

Abbildung 4.37 zeigt im linken Graph die Ergebnisse der Gap-Statistik-Methode. Es ist zu sehen, dass die ermittelte optimale Anzahl an Clustern mit der Anzahl der unterschiedlichen Typen von Instanzen übereinstimmt. Damit liefert die Untersuchung einen ersten Hinweis dafür, dass die verschiedenen Typen von Instanzen sich anhand der eingeführten Eigenschaften unterscheiden lassen. Mithilfe der bestimmten Anzahl von Clustern kann der K-Means-Clusteralgorithmus auf die transformierten Eigenschaften angewendet werden. Abbildung 4.37 zeigt im rechten Graph das Ergebnis des Clusterings. Es ist zu sehen, dass der Algorithmus zuverlässig die vier Cluster korrekt zuordnet. Die leichten Überschneidungen der Cluster 1 (Typ A) und 3 (Typ C) resultieren unter anderem aus der Projektion von dem mehrdimensionalen Eigenschaftenraum hin zu einem zweidimensionalen Raum für die Visualisierung. Die Dimensionsreduktion wird mit der Hauptkomponentenanalyse durchgeführt. Hierbei werden eine Vielzahl von Eigenschaften durch eine geringere Zahl an Linearkombinationen dieser angenähert.

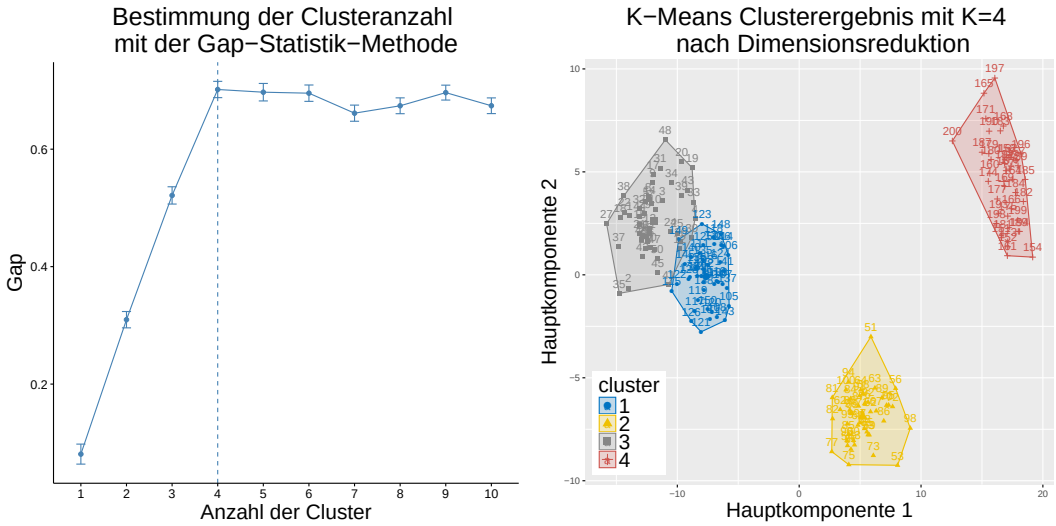

Abbildung 4.37: Ergebnisse der Gap-Statistik-Methode (linker Graph) und des K-Means Clusterings mit K=4 (rechter Graph)

Der Rand-Index R, eingeführt in Rand (1971), ist ein Maß für die Ähnlichkeit zweier Clusterergebnisse. Der Index berechnet sich aus dem Quotienten aus der Summe der gleich gruppierten Objektpaare a und ungleich gruppierten Objektpaare b und dem Binominialkoeffizienten. Die Berechnung des Rand-Indexes R erfolgt nach Gleichung 4.13, wobei n die Anzahl aller zu clusternden Objekte ist. Beträgt der Rand-Index $R = 1$, sind die Clusterergebnisse identisch. Jedes zufällig gewählte Beispiel ist in den Ergebnissen in das gleiche Cluster eingeordnet.

$$R = \frac{a+b}{\binom{n}{2}} \tag{4.13}$$

Der Rand-Index für das K-Means-Clusterergebnis und der tatsächlichen Zuordnung

von DVRP-Eigenschaften auf die Typen aus Abbildung 4.2 beträgt $R = 0.96$. Das bedeutet, dass eine zufällig gewählte Probleminstanz zu 96% mithilfe der in dieser Arbeit eingeführten Eigenschaften dem richtigen Typen zugeordnet werden kann. Die Problemeigenschaften bilden also eine solide Grundlage, um mithilfe einer einfachen Clusteranalyse die unterschiedlichen Typen der Probleminstanzen klar zu identifizieren.

Um die signifikanten Eigenschaften für das Clustering zu bestimmen wird der Datensatz mithilfe eines Entscheidungsbaums untersucht. Der in Abbildung 4.38 abgebildete Baum ist mit der *Conditional Inference Trees* (CTREE)-Methode, entwickelt von Hothorn et al. (2006), konstruiert. Der binäre Entscheidungsbaum wird durch schrittweises Aufspalten der Menge der Probleminstanzen mithilfe der eingeführten Eigenschaften rekursiv erstellt. In einem ersten Schritt wird eine Hypothese über die Unabhängigkeit zwischen den für die Teilmenge relevanten Eigenschaften und den Typen der Instanzen aufgestellt. Wird die Hypothese nicht verworfen, stoppt der rekursive Algorithmus und die bearbeitete Teilmenge wird nicht weiter unterteilt. Wird die Hypothese hingegen verworfen, wird die Eigenschaft für das Aufspalten der Teilmenge ausgewählt, die die stärkste Assoziation zu den Typen der Instanzen hat. Für die Bestimmung der Assoziation werden die p-Werte des Ergebnisses der Hypothesentests herangezogen. Im zweiten Schritt wird die Teilmenge mithilfe der ausgewählten Eigenschaften in zwei disjunkte Teilmengen zerlegt. Auch hier werden für alle möglichen Teilmengenkombinationen Teststatistiken, die den Unterschied der Teilmengen erfassen, berechnet. Die Teilmengenkombination mit dem größten Unterschied ist das Ergebnis der Aufspaltung. Die beiden Schritte werden rekursiv für die neu entstehenden Teilmengen angewandt bis keine Unterteilung mehr möglich ist. Eine Unterteilung ist dann nicht mehr möglich, wenn die Hypothese über die Unabhängigkeit nicht verworfen werden kann.

Die Untersuchung mithilfe des Entscheidungsbaums bestätigt die Ergebnisse der visuellen Analyse der Wertebereiche. Klar sieht man in Abbildung 4.38, dass mit der Fläche- und Schwerpunkt-Eigenschaft die Typen A und B eindeutig voneinander unterschieden werden können. Die Typen C und D unterscheiden sich anhand der Flächen- und Winkel-Eigenschaften mit der Nachbarschaft N mit $n = 5$ Nachbarn. Mithilfe dieser drei Eigenschaften aus der Kategorie der Eigenschaften, die die Beziehungen zwischen dynamischen und statischen Anfragen beschreiben, lassen sich alle Typen von Instanzen eindeutig unterscheiden. Die eingeführten DVRP-Eigenschaften erweisen sich als gute Repräsentanten um DVRP-Instanzen mit identischem DOD voneinander zu unterscheiden und können damit einen wesentlichen Beitrag für das Verständnis zwischen dem Zusammenhang von Algorithmenperformance und Beschaffenheit bzw. Aufbau einer Probleminstanz leisten. Die neu eingeführten Eigenschaften bilden so eine solide Grundlage für die automatisierte Auswahl von Algorithmen. Im Kontext des ASP bildet die Gesamtheit der Eigenschaften für das DVRP den Eigenschaftenraum F (vgl. Abbildung 2.9).

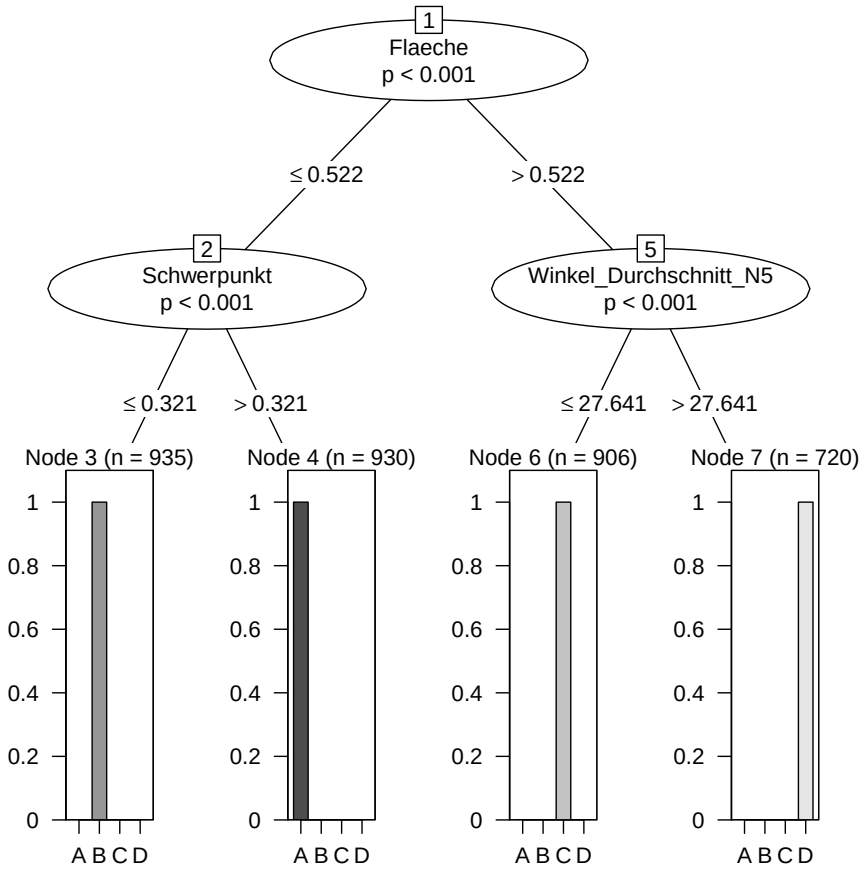

Abbildung 4.38: Klassifikationsregeln dargestellt mithilfe eines Entscheidungsbaums konstruiert mit der CTREE Methode für die Klassifikation der Instanztypen A, B, C und D

4.1.5 Zusammenfassung und Diskussion

In den vorherigen Abschnitten werden 181 verschiedene DVRP-Eigenschaften eingeführt. Zehn davon sind bereits bekannte TSP-Eigenschaften, die auf die Menge der dynamischen Kundenfrage in einem DVRP angewendet werden. Die restlichen 171 beschreiben die Beziehung zwischen statischen und dynamischen Kundenanfragen, um so die Dynamik einer Probleminstanz zu charakterisieren. Die Performance von Algorithmen kann entscheidend von der Dynamik in einer Probleminstanz beeinflusst werden (vgl. Larsen, 2000, Mayer et al., 2017, Mayer et al., 2018*a* oder Mayer et al., 2018*b*).

Die Wertebereiche aller Problemeigenschaften werden mithilfe von vier unterschiedlichen Typen von Probleminstanzen analysiert. Neben den vier betrachteten Instanztypen gibt es eine Vielzahl von möglichen Strukturen die dynamische und statische Kundenanfragen bilden können (vgl. Fränti und Sieranoja, 2018). Trotz der Beschränkung auf die vier Typen ergeben die Ergebnisse der Analyse einen sehr guten Eindruck über die Ausprägungen der Eigenschaften bei bestimmten Beschaffenheiten der Instanzen. Der Zusammenhang zwischen der Ausprägung einer Eigenschaft und der Beschaffenheit

einer Probleminstanz wird zusätzlich mithilfe von gezielt generierten Probleminstanzen erforscht. Die Basis für die generierten Instanzen ist eine sehr gleichmäßige VRP-Instanz (vgl. Abbildung 4.3). Auch wird immer genau die Hälfte aller Anfragen für dynamische Anfragen durch den GA ausgewählt. Trotz dieser Einschränkungen vermitteln die generierten Instanzen einen guten Eindruck von eigenschaftenspezifischen Strukturen, die dynamische und statische Anfragen in einer DVRP-Instanz bilden. Die abschließend durchgeführte Evaluation bestätigt die Eignung der Eigenschaften zum Unterscheiden von Probleminstanzen.

Die hier neu eingeführten Problemeigenschaften ergänzen die in Unterabschnitt 2.1.3 erläuterten Eigenschaften bei der Charakterisierung von DVRP-Instanzen. Damit wird es erstmals möglich eine Vielzahl von Probleminstanzen voneinander abzugrenzen und anhand von Gemeinsamkeiten oder Unterschieden zu gruppieren. Nur so ist ein fairer Vergleich von Algorithmen gewährleistet. Die neu eingeführten Eigenschaften ermöglichen ebenfalls die zielgerichtete Generierung von Probleminstanzen, die sich signifikant voneinander unterscheiden. Das erlaubt eine bessere Analyse des Verhaltens von Algorithmen und zusätzlich wird so eine zielgerichtete Konstruktion von Algorithmen ermöglicht. Die Unterscheidbarkeit von Problemstellungen ist zudem eine Grundvoraussetzung für die erfolgreiche Selektion von Algorithmen. Zu beachten ist, dass die entwickelten Eigenschaften auch Statistiken über die dynamischen Kundenanfragen einbeziehen. Oftmals ist aber in einer realen Problemstellung über genau diese Anfragen nur wenig bekannt. Bei der Umsetzung der automatisierten Auswahl von Algorithmen werden die Problemeigenschaften identifiziert, die einen wesentlichen Einfluss auf die Performance von Routing Algorithmen in einem dynamischen Umfeld haben. Abschnitt 5.2 beschäftigt sich intensiv mit den Beziehungen zwischen Algorithmen und Eigenschaften und identifiziert auch eine minimale Menge von wichtigsten Problemeigenschaften, die für die automatisierte Selektion von Algorithmen notwendig ist. Es werden also die wesentlichen Charakteristika von Probleminstanzen identifiziert, die bei einer Prognose über zukünftigen Anfragen berücksichtigt werden müssen, um geeignete Algorithmen für DVRP auszuwählen.

4.2 Lösungsansätze für Dynamische Routingprobleme

Eine wichtige Voraussetzung für die automatisierte Selektion von Algorithmen ist mit der Einführung von Eigenschaften für das DVRP in Abschnitt 4.1 erfüllt. Für die Evaluation der Problemeigenschaften und für die Umsetzung der automatisierten Selektion von Algorithmen müssen verschiedene Lösungsansätze für das DVRP umgesetzt werden. Allgemein werden in der Literatur Lösungsansätze basierend auf verschiedenen Methoden beschrieben (vgl. Unterabschnitt 2.1.5). Oft konzentrieren sich konkrete Implementierungen auf Problemstellungen mit speziellen Nebenbedingungen. Allgemein gilt, dass sich ein Lösungsansatz für ein DVRP, bei dem nicht alle Kundenanfragen

dynamisch sind, aus mindestens zwei Teilen zusammensetzt (vgl. Unterabschnitt 2.1.5). Der Planungsteil oder Planungsalgorithmus plant Routen für alle statischen Kundenanfragen. Der Anpassungsteil oder Anpassungsalgorithmus reagiert auf die sich ändernden Probleminformationen und plant dynamische Kundenanfragen in bereits bestehende Routen oder kreiert neue. Dabei stehen dem Anpassungsalgorithmus oft Ergebnisse des Planungsalgorithmus zur Verfügung. Beide Algorithmen können aber auch unabhängig voneinander arbeiten.

In einer DVRP-Problemstellung mit einem $DOD < 1$ und ohne explizites Vorwissen über dynamische Kundenanfragen ist das vom Planungsalgorithmus zu lösende Problem ein VRP. Als Eingaben stehen die folgenden Parameter zur Verfügung.

- Menge der Knoten $V = \{v_0, v_1, ..., v_n\}$, repräsentieren die zu beliefernden, statischen Kunden
- Menge der zu liefernden Gütern q_i für alle Kunden in V
- Menge der Kanten $E = \{(v_i,v_j)|v_i,v_j \in V; i \neq j\}$, beschrieben durch die Menge der benachbarten Kunden
- Kosten $c_{i,j}$ zwischen den Kunden v_i und v_j, definiert in einer Kostenmatrix C
- Anzahl m der zur Verfügung stehenden Fahrzeuge und die zugehörige Transportkapazität k

In der in dieser Arbeit betrachteten Problemstellungen ist die Menge der zu liefernden Güter $q_i = 1$ für alle Kundenanfragen. Es wird zusätzlich nur ein Fahrzeug mit einer unendlichen Transportkapazität k angenommen (vgl. Abschnitt 2.3). Unter den getroffenen Annahmen reduziert sich das betrachtete Problem für einen Planungsalgorithmus auf ein TSP, weil $\sum_0^n q_i < k$. Der Algorithmus muss also genau eine Rundreise R_i für das Fahrzeug i planen. Das Ziel der Planung ist es eine Route zu finden, bei der die zurückgelegte Distanz minimal ist. Die geplante Tour muss alle statischen Kundenanfragen als Zwischenstationen enthalten.

Die vom Planungsalgorithmus berechnete Rundreise R_i steht neben dem einzuplanenden dynamischen Kunden v_x als Eingabe für den Anpassungsalgorithmus zur Verfügung. Dem Algorithmus steht es frei den zu planenden Kunden v_x in die bestehende Tour zu integrieren („partielle Planrevision“) oder eine neue Tour für alle noch nicht bedienten Kunden zu planen („totalen Planrevision“ für „voll disponible“ Kunden). In der vorliegenden Arbeit werden beide Ansätze betrachtet. Ein Anpassungsalgorithmus kann grundsätzlich die bereits geplanten Touren in jedem möglich Zustand verändern, also auch dann, wenn sich Fahrzeuge zwischen zwei Kundenanfragen befinden. Die Arbeiten von Ichoua et al. (2000), Regan et al., 1995 und Regan et al., 1996, zum Beispiel, beschäftigen sich mit dem Rerouting von Fahrzeugen zwischen Kunden. Hier wird eine Fahrt von einer Anfrage zu einer anderen umgeleitet. Alle in dieser Arbeit betrachteten Anpassungsalgorithmen berücksichtigen nicht das Umleiten von Fahrzeugen.

Ein Algorithmus für das DVRP kann grundsätzlich zwei verschiedene Ansätze zur Lösungsfindung verfolgen (vgl. Unterabschnitt 2.1.5). Ein erster Ansatz plant eine nahe optimale Rundreise für das TSP. Dabei handelt es sich um einen Algorithmus der Kategorie Re-Optimierung (RO) (vgl. Unterabschnitt 2.1.5). Ein zweiter Ansatz plant eine strategische, robuste Rundreise, die größere Flexibilität für mögliche dynamische Kundenanfragen bietet. Der zweite Ansatz berücksichtigt also implizit Wissen über dynamische Kundenanfragen. Dieser Algorithmus ist in die Kategorie *Policy Function Approximation* (PFA) einzuordnen (vgl. Unterabschnitt 2.1.5). Beide Ansätze sind in der Praxis weit verbreitet und werden aus diesem Grund in der vorliegenden Arbeit betrachtet. Algorithmen, die Warte- oder Platzierungsstrategien verfolgen, werden nicht berücksichtigt.

In den folgenden Unterkapiteln werden Algorithmen motiviert und vorgestellt, die in verschiedenen Kombinationen in dieser Arbeit für die Evaluation der eingeführten Eigenschaften und für die Umsetzung der automatisierten Selektion von Algorithmen verwendet werden. Dabei ist der Spiralalgorithmus im Zuge der vorliegenden Arbeit entwickelt worden. In Unterabschnitt 4.2.8 werden dann die konkreten Variationen der erläuterten Algorithmen, die für die Lösung des DVRP herangezogen werden, aufgezeigt.

4.2.1 Nächster-Nachbar-Heuristik

Die Nächster-Nachbar (NN)-Heuristik ist ein gieriges Konstruktionsverfahren für Lösungen für das TSP (Lawler, 1985). Ausgehend von einem aus der Menge aller Knoten $V = \{v_0, v_1, ..., v_n\}$ zufällig gewählten Startknoten v_s wird dabei der Folgeknoten v_{s+1} aus der Menge $V \setminus \{v_s\}$ so bestimmt, dass die Distanz $c_{s,s+1}$ zwischen v_s und v_{s+1} minimal ist. Sukzessive werden so alle Knoten aus V in die Rundreise geplant. Die NN-Heuristik vernachlässigt bei der Konstruktion von Lösungen den Abstand zwischen Start- und Endknoten. So können beliebig weit vom Optimum entfernte Rundreisen für ein TSP geplant werden (Lawler, 1985). Die Nächster-Nachbar-Heuristik ist in $O(n^2)$. Abbildung 4.39 visualisiert einen Konstruktionsschritt der NN-Heuristik anhand eines einfachen Beispiels. Für die Bestimmung des Nachfolgers von Knoten A werden die Kosten $c_{A,D}$, $c_{A,C}$ und $c_{A,B}$ bestimmt. Im Beispiel aus Abbildung 4.39 entsprechen die Kosten der Distanz zwischen den Knoten. Der Knoten B ist am nächsten zu A und wird aus diesem Grund als Nachbar von A in die Rundreise geplant (vgl. rechtes Bild in Abbildung 4.39).

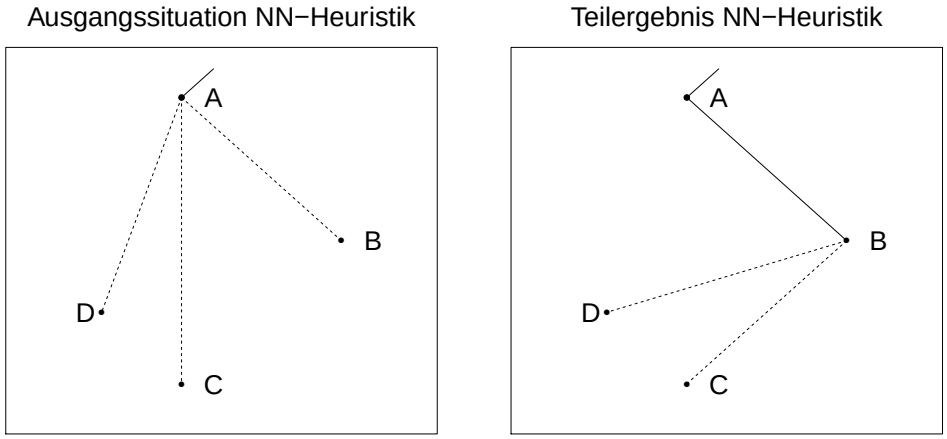

Abbildung 4.39: Konstruktionsschritt der Nächster-Nachbar (NN)-Heuristik (der Knoten B wird als Nachfolger des Knotens A in die Rundreise geplant, weil dieser die geringsten Kosten $c_{B,A}$ verursacht)

4.2.2 Insert-Operator

Der Insert-Operator fügt einen noch nicht geplanten Knoten v_x zwischen zwei benachbarte, bereits geplante Knoten v_{x-1} und v_{x+1} ein. Der Knoten v_x wird dabei zwischen die Nachbarn geplant, für die die zusätzlich entstehenden Kosten $c_{x,x-1,x+1}$ minimal sind. Die Berechnung von $c_{x,x-1,x+1}$ ist in Gleichung 4.14 erfasst. Der Insert-Operator kann sowohl für die Konstruktion, als auch für die Verbesserung einer existierenden Lösung verwendet werden. Die einmalige Anwendung des Insert-Operators ist in $O(n)$. Für das Einplanen eines Knotens werden die zusätzlich entstehenden Kosten zwischen allen Nachbarn in der Route R ausgewertet. Abbildung 4.40 visualisiert die Anwendung des Insert-Operators anhand eines Beispiels. Für das Planen des noch ungeplante Knoten D werden die zusätzlich entstehenden Kosten $c_{D,A,B} = c_{A,D} + c_{B,D} - c_{A,B}$ und $c_{D,B,C} = c_{B,D} + c_{C,D} - c_{B,C}$ ausgewertet. Da $c_{D,A,B} > c_{D,B,C}$ wird der Knoten D zwischen die Knoten B und C geplant.

$$c_{x,x-1,x+1} = c_{x-1,x} + c_{x,x+1} - c_{x-1,x+1} \tag{4.14}$$

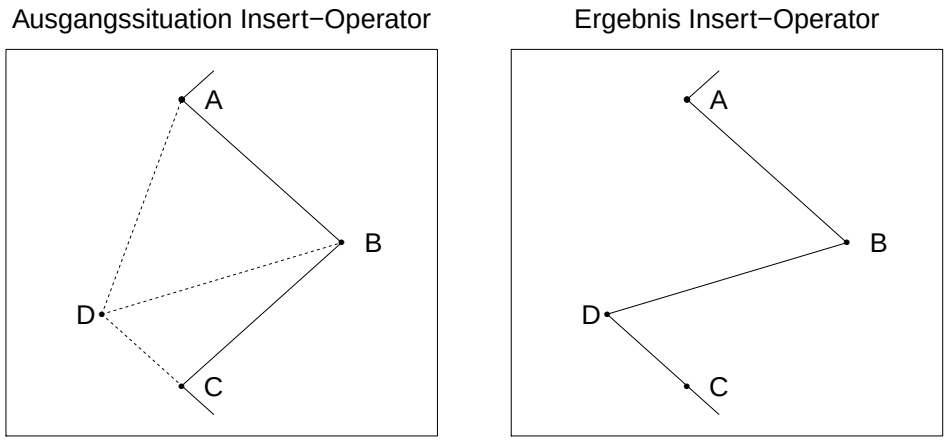

Abbildung 4.40: Arbeitsschritt des Insert-Operators (Knoten D wird zwischen B und C geplant)

4.2.3 2-Opt-Operator

Der 2-Opt-Operator ist ein einfacher lokaler Suchalgorithmus für die Konstruktion und die Verbesserung von Lösungen für das TSP, eingeführt von Croes (1958). Die Grundidee des Algorithmus ist das Auflösen von sich kreuzenden Verbindungen zwischen benachbarten Knoten in einer Rundreise. Dazu werden zwei beliebige Knoten v_i und v_k aus allen n Knoten der Rundreise für eine Vertauschung (engl. swap) ausgewählt. Eine neue Reise R_{neu} wird aus den Teilstrecken $T_1 = \{v_0, \ldots, v_{i-1}\}$, $T_2 = \{v_i, \ldots, v_k\}$ und $T_3 = \{v_k + 1, \ldots, v_n\}$ nach $R_{neu} = T_1 + reverse(T_2) + T_3$ konstruiert. Die Funktion $reverse$ kehrt dabei die Reihenfolge der Knoten in T_2 um, $reverse(T_2) = \{v_k, \ldots, v_i\}$. Der 2-Opt-Algorithmus ist in $O(n^2)$, da für die durchgeführten Vertauschung alle n Knoten aus V bis zur Terminierung des Algorithmus miteinander kombiniert werden (Bentley, 1992). Abbildung 4.41 zeigt die Anwendung des 2-Opt-Operators anhand eines einfachen Beispiels. Die Knoten C und D sind für die Vertauschung ausgewählt. Die Teilstrecken ergeben sich wie folgt: $T_1 = \{\ldots, A\}$, $T_2 = \{C, \ldots, D\}$ und $T_3 = \{B, \ldots\}$. Die neue Reise ergibt sich aus $R_{neu} = T_1 + reverse(T_2) + T_3 = \{\ldots, A, D, \ldots, C, B, \ldots\}$.

Weiterentwicklungen des 2-Opt-Operators sind k-Opt-Operatoren, die Vertauschungen für k Knoten aus V beschreiben. K-Opt-Operatoren sind in $O(n^k)$. Die Lin-Kernighan-Heuristik eingeführt in Lin und Kernighan (1973) passt die Anzahl der k Knoten, die während einer Optimierung vertauscht werden, dynamisch an und reduziert so die Laufzeit gegenüber k-Opt-Operatoren signifikant, erreichen aber nicht die Laufzeit des 2-Opt-Operators. Trotz genannter Weiterentwicklungen ist der 2-Opt-Operator für große Probleminstanzen in der Praxis sehr erfolgreich und weit verbreitet (vgl. Mersmann et al., 2012). Aus diesem Grund und da im Zuge der vorliegenden Arbeit sehr viele und große Probleminstanzen gelöst werden müssen, wird der 2-Opt-Operator gegenüber der Lin-Kernighan-Heuristik bevorzugt.

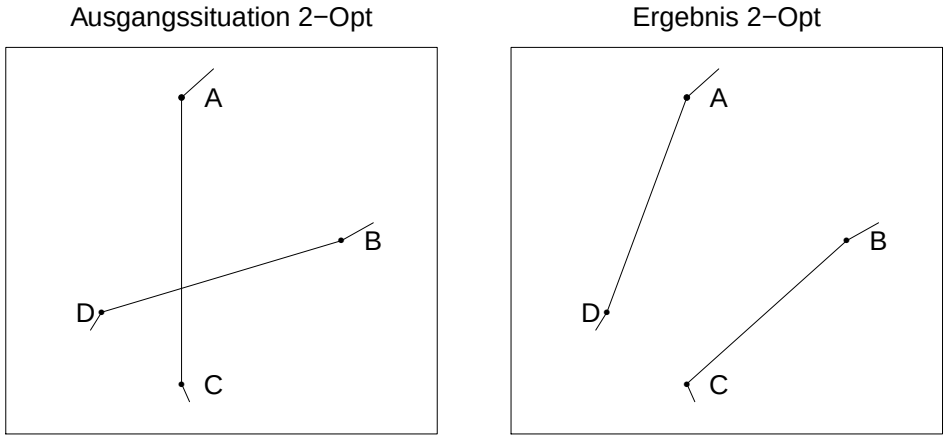

Abbildung 4.41: Auflösung der Kreuzung der Kanten durch die Anwendung des 2-Opt-Operators

4.2.4 Stringing-Operatoren

Die in dieser Arbeit verwendeten Stringing-Operatoren wurden im Zuge der *Generalized Insertion Procedure* (GENI) von Gendreau et al. (1992) entwickelt. GENI ist ein Konstruktionsverfahren für Lösungen für das TSP. Der Algorithmus fügt ausgehend von einer Rundreise mit drei zufällig gewählten Knoten alle restlichen Knoten der Menge V mithilfe der Stringing-Operatoren hinzu. Ausgehend von einer Menge beliebig gewählter Knoten V definieren beide Operatoren eine Menge zu löschender Kanten, eine Menge neu einzufügender Kanten und eine Menge von Teilstrecken, deren Knotenreihenfolge angepasst werden muss. Abbildung 4.42 veranschaulicht das Vorgehen des Stringing-Operators vom Typ 1 bei dem Einfügen des Knoten v_x in die bestehende Rundreise. Die Knoten v_i, v_j, und v_k werden aus der Menge V so gewählt, dass $v_i \neq v_k$ und $v_j \neq v_k$. Nach der Wahl der Knoten werden die Kanten (v_i, v_{i+1}), (v_j, v_{j+1}), und (v_k, v_{k+1}) aus der bestehenden Rundreise entfernt. Im Anschluss werden die Kanten (v_i, v_x), (v_x, v_j), (v_{i+1}, v_k) und (v_{j+1}, v_{k+1}) der Rundreise hinzugefügt und die Reihenfolge der Knoten in den Teilstrecken $T_1 = \{v_{i+1}, \ldots, v_j\}$ und $T_2 = \{v_{j+1}, \ldots, v_k\}$ umgekehrt.

Der Stringing-Operator vom Typ 2 ist in Abbildung 4.43 veranschaulicht. Hier werden die Knoten v_i, v_j, v_k und v_l so gewählt, dass $v_j \neq v_k$, $v_{j+1} \neq v_k$, $v_i \neq v_l$ und $v_{i+1} \neq v_l$. Nach der Wahl der Knoten werden die Kanten (v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1}), und (v_{k-1}, v_k) aus der bestehenden Rundreise entfernt. Die Kanten (v_i, v_x), (v_x, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) und (v_{i+1}, v_k) werden der Rundreise hinzugefügt und die Reihenfolge der Knoten in den Teilstrecken $T_1 = \{v_{i+1}, \ldots, v_{l-1}\}$ und $T_2 = \{v_l, \ldots, v_j\}$ werden umgekehrt.

Aufgrund der möglich Kombinationen von v_i, v_j, v_k und v_l liegen die Stringing-Operatoren in $O(n^4)$. Um die Laufzeit der Operatoren zu verkürzen, werden die möglichen Kombinationen der Knoten mithilfe von Nachbarschaftsbeziehungen eingeschränkt. Für jeden Knoten v_i in V wird die p-Nachbarschaft $N_p(v_i)$ bestimmt. In $N_p(v_i)$ sind die p Knoten, die v_i am nächsten in Bezug auf die Kostenmatrix C sind. Die Knoten v_i, v_j,

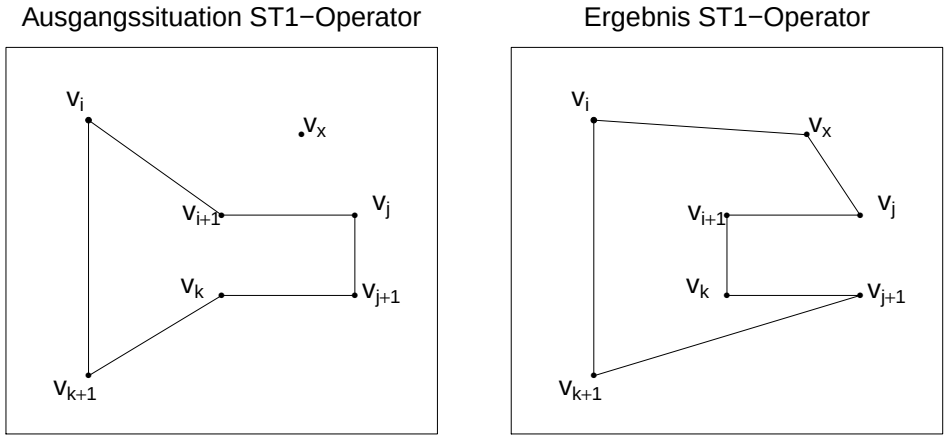

Abbildung 4.42: Einfügen des Knotens v_x zwischen die Knoten v_i und v_j durch den Stringing-Operator vom Typ 1

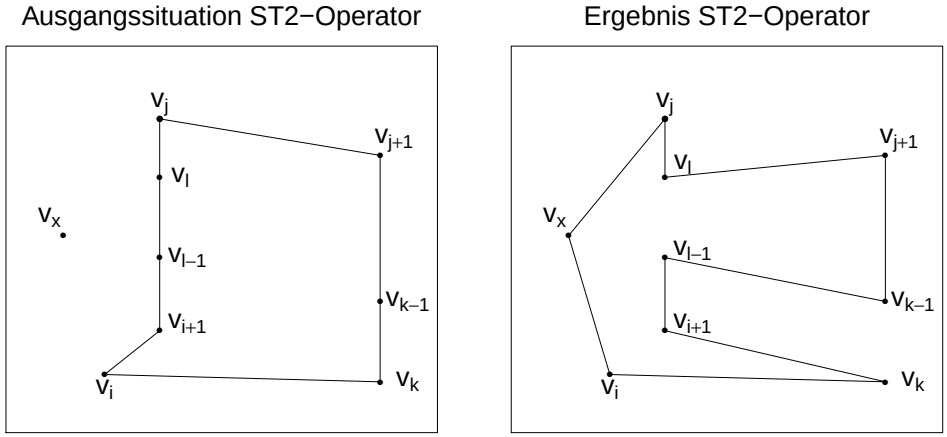

Abbildung 4.43: Einfügen des Knotens v_x zwischen die Knoten v_i und v_j durch den Stringing-Operator vom Typ 2

v_k und v_l werden jetzt so gewählt, dass $v_i, v_j \in N_p(v_x)$, $v_k \in N_p(v_{i+1})$ und $v_l \in N_p(v_{j+1})$.

Der GENI-Algorithmus konstruiert eine Lösung für das TSP, indem alle Knoten aus V einer beliebigen Startlösung bestehend aus drei Knoten mithilfe der Stringing-Operatoren sukzessive hinzugefügt werden. Bei jedem Hinzufügen werden beide Operatoren und beide mögliche Startrichtungen der Rundreise betrachtet. Es wird der Operator mit der Richtung angewandt, der die wenigstens Mehrkosten durch das Einfügen verursacht.

4.2.5 Unstringing-Operatoren

Die in dieser Arbeit verwendeten Unstringing-Operatoren wurden auch im Zuge des GENI-Algorithmus von Gendreau et al. (1992) eingeführt und bilden die Umkehroperationen der Stringing-Operatoren. Der Unstringing-Operator vom Typ 1 ist das Gegenstück zum Stringing-Operator vom Typ 1 und der Unstringing-Operator vom

Typ 2 ist das Gegenstück zum Stringing-Operator vom Typ 2. Auch die Unstringing-Operatoren definieren eine Menge zu löschender Kanten, eine Menge neu einzufügender Kanten und eine Menge von Teilstrecken, deren Knotenreihenfolge angepasst werden muss. Abbildung 4.44 visualisiert das Vorgehen des Unstringing-Operators vom Typ 1 beim Entfernen des Knotens v_i aus der Rundreise. Nach der Auswahl der Knoten v_j und v_k, wobei $v_j \in N_p(v_{i+1})$ und $v_k \in N_p(v_{i-1})$ ist, werden die Kanten (v_{i-1}, v_i), (v_i, v_{i+1}), (v_k, v_{k+1}) und (v_j, v_{j+1}) gelöscht. Die gelöschten Kanten werden durch die Kanten (v_{i-1}, v_k), (v_{i+1}, v_j) und (v_{k+1}, v_j) ersetzt. Nach dem Umkehren der Reihenfolge der Knoten in den Teilstrecken $T_1 = \{v_{i+1}, \ldots, v_k\}$ und $T_1 = \{v_{k+1}, \ldots, v_j\}$ ist die Rundreise ohne v_i konstruiert.

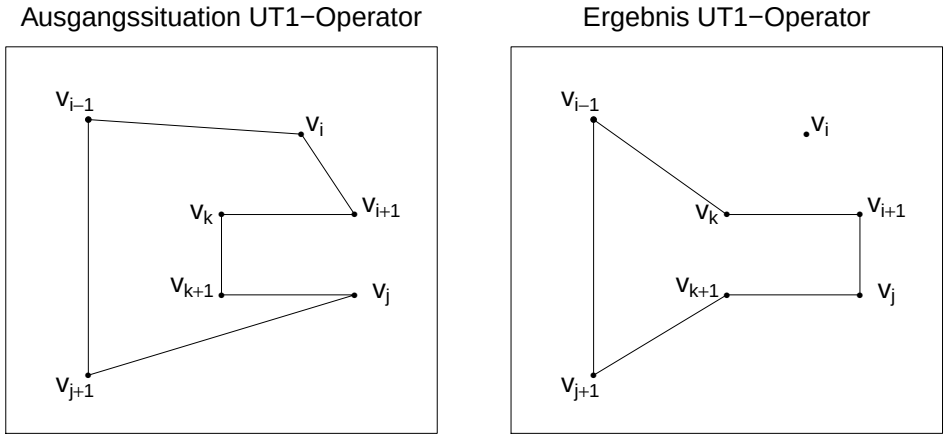

Abbildung 4.44: Entfernen des Knotens v_i aus der Rundreise durch den Unstringing-Operator vom Typ 1

Abbildung 4.45 visualisiert das Vorgehen des Unstringing-Operators vom Typ 2 beim Entfernen des Knotens v_i aus der Rundreise. Die Knoten v_j, v_k und v_l werden hier so gewählt, dass $v_j \in N_p(v_{i+1})$, $v_k \in N_p(v_{i-1})$ und $v_l \in N_p(v_{k+1})$ ist. Zusätzlich gilt für den Knoten v_k, dass dieser Bestandteil der Teilstrecke $T_1 = \{v_{j+1}, \ldots, v_{i-2}\}$ und das v_l Bestandteil der Teilstrecke $T_2 = \{v_j, \ldots, v_{k-1}\}$ ist. Nach dem Löschen der Kanten (v_{i-1}, v_i), (v_i, v_{i+1}), (v_{j-1}, v_j), (v_l, v_{l+1}) und (v_k, v_{k+1}) werden die Kanten (v_{i-1}, v_k), (v_{l+1}, v_{j-1}), (v_{i+1}, v_j) und (v_l, v_{k+1}) eingefügt. Nach dem Umkehren der Reihenfolge der Knoten in den Teilstrecken $T_1 = \{v_{i+1}, \ldots, v_{j-1}\}$ und $T_1 = \{v_{l+1}, \ldots, v_k\}$ ist die Rundreise ohne v_i konstruiert.

Die Laufzeit der Unstringing-Operatoren ist identisch mit der Laufzeit der Stringing-Operatoren und kann durch das Betrachten einer eingeschränkten Nachbarschaft reduziert werden. In Kombination mit den Stringing-Operatoren bilden die Unstringing-Operatoren einen Optimierungsalgorithmus, eingeführt als *Unstringing/Stringing* (US) Algorithmus in Gendreau et al. (1992), der bestehende Rundreisen für das TSP kontinuierlich verbessert. Dazu werden alle Knoten v in V mit beiden Unstringing-Operatoren entfernt und mit den Stringing-Operatoren erneut hinzugefügt. Die Änderung der Rundreise wird übernommen, wenn eine Verbesserung erzielt wird.

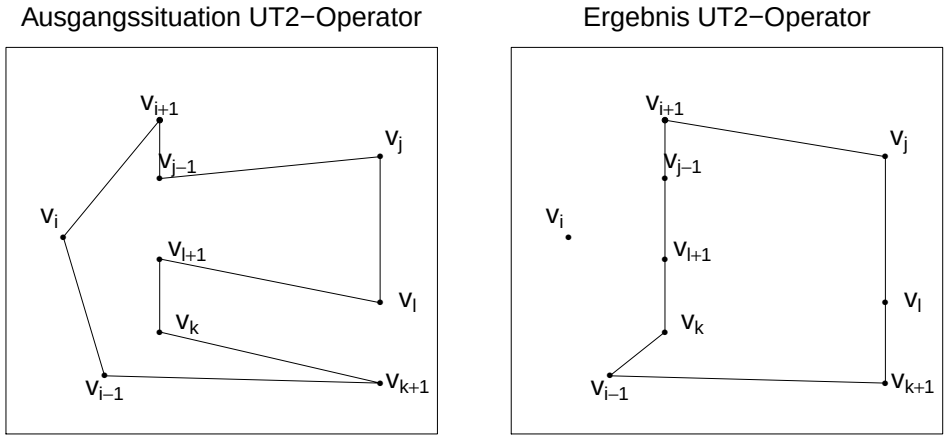

Abbildung 4.45: Entfernen des Knotens v_i aus der Rundreise durch den Unstringing-Operator vom Typ 2

Der US-Algorithmus ist ein weit verbreiteter Optimierungsalgorithmus, der nach wie vor in aktuellen Lösungsansätzen für das TSP verwendet wird (vgl. Mavrovouniotis et al., 2017).

4.2.6 Min-Max-Ant-System

Ant Colony Optimization (ACO) Heuristiken besten aus einer Population von μ Ameisen (engl. ants). Diese Ameisen konstruieren pro Iteration eine Lösung für Optimierungsprobleme und tauschen untereinander Informationen über gefundene Lösungen über Iterationen hinweg mithilfe von Pheromonspuren aus. Der erste entwickelte ACO-Algorithmus ist ein *Ant System* (AS) beschrieben von Colorni et al. (1991) für die Konstruktion von Lösungen für das TSP (Mavrovouniotis et al., 2015). Verschiedene AS für die unterschiedlichsten Anwendungen sind seit dem entwickelt worden (Dorigo und Stützle, 2003). Besonders erfolgreich beim Lösen von TSP ist die AS Variation *Max Min AS* (MMAS) (Stutzle und Hoos, 1997). MMAS zeichnen sich durch teilweise dynamische Maximum- τ_{max} und Minimumwerte τ_{min} für Pheromone aus. Mithilfe von Pheromonen, die zum Erzeugen und zum Speichern von Lösungen verwendet werden, werden Informationen über Lösungen zwischen Ameisen über Iterationen hinweg kommuniziert.

Der in dieser Arbeit verwendete MMAS-Algorithmus orientiert sich an den Arbeiten von Mavrovouniotis et al. (2015) und Mavrovouniotis et al. (2017). Jede Ameise k konstruiert eine eigene Lösung für das TSP. Dabei wird der als nächstes zu besuchende Knoten v_j mithilfe der probabilistischen Regel, angegeben in Gleichung 4.15, ausgewählt. Der Knoten v_j beschränkt sich auf noch nicht besuchte, der p-Nachbarschaft $N_p^k(v_i)$ angehörige Knoten.

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_p^k(v_i)} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, \text{ if } j \in N_p^k(v_i) \tag{4.15}$$

Die Wahrscheinlichkeit p_{ij}^k, dass die Ameise k den Knoten v_j als Nachfolger von v_i wählt, berücksichtigt den aktuellen Pheromonwert τ_{ij} und die heuristische Information η_{ij} für den Übergang von v_i nach v_j. Die heuristische Information zwischen den Knoten v_i und v_j wird in Routingproblemstellungen üblicherweise mithilfe der Kosten c_{ij} zwischen den Knoten mit $\eta_{ij} = 1/c_{ij}$ bestimmt. Mit den Parametern α und β lassen sich die Einflüsse des Pheromonwerts und der heuristischen Information anpassen.

Alle Pheromonwerte werden unabhängig von Lösungen von Ameisen pro Iteration einem Verdunstungsprozess ausgesetzt. Eine Iteration umfasst die Konstruktion einer vollständigen Lösung pro Ameise. Der Verdunstungsprozess ist in Gleichung 4.16 angegeben. Die Verdunstungsrate p liegt im Intervall [0,1] und kann problemspezifisch konfiguriert werden. Für τ'_{ij} gilt, dass $\tau'_{ij} \geq \tau_{min}$.

$$\tau'_{ij} \leftarrow (1-p)\tau_{ij}, \forall(v_i, v_j) \tag{4.16}$$

Nachdem der Verdunstungsprozess durchgeführt ist, werden die Pheromonwerte nach Gleichung 4.17 aktualisiert.

$$\tau''_{ij} \leftarrow \tau'_{ij} + \triangle\tau_{ij}^{best}, \forall(v_i, v_j) \in T^{best} \tag{4.17}$$

Der neue Pheromonwerte τ''_{ij} addiert dem ursprünglichen Pheromonwert τ'_{ij} das Delta $\triangle\tau_{ij}^{best}$, wobei $\tau''_{ij} \leq \tau_{max}$ gilt. Das Delta ergibt sich dabei aus $\triangle\tau_{ij}^{best} = 1/C^{best}$. C^{best} ist die Lösungsqualität der besten durch eine Ameise gefunden Rundreise T^{best}. Abwechselnd entspricht C^{best} der Qualität der besten je gefunden Rundreise C^{bs} (bs - best so far) und der Qualität der besten Rundreise einer Iteration C^{ib} (ib - iteration best). Alle Pheromonwerte werden mit τ_{min} initialisiert.

Die Grenze für das Maximum der Pheromonwerte τ_{max} wird nach jeder Iteration nach $1/pC^{bs}$ aktualisiert. Der initiale Wert für C^{bs} ist das geschätzte Optimum für die Lösung des TSP.

Der MMAS-Algorithmus ist bei μ Ameisen, n Knoten und i Iterationen in $O(i * n^2 * \mu)$. Mavrovouniotis et al. (2015) ergänzen den beschriebenen MMAS-Algorithmus um eine lokale Suche. Jede beste Lösung, erzeugt in einer Iteration, wird mithilfe des US-Algorithmus nachträglich optimiert. Die zusätzlich lokal optimierte Lösung bildet die Grundlage für die Aktualisierung der Pheromonwerte τ''_{ij} und der Maximumgrenze τ_{max}. Der in dieser Arbeit verwendete MMAS-Ansatz optimiert nicht jede beste Lösung einer Iteration lokal. Vielmehr wird die finale Rundreise mit dem US-Algorithmus verbessert.

4.2.7 Spiralalgorithmus

Der Spiralalgorithmus ist ein Konstruktionsverfahren für Lösungen für das TSP. Das Verfahren wird im Zuge der vorliegenden Arbeit entwickelt. Dabei verfolgt der Spiralansatz nicht das Ziel eine nahe optimale Rundreise für das TSP zu planen, vielmehr wird versucht, eine möglichst robuste und damit flexible Route zu kreieren, um auf mögliche dynamische Kundenanfragen gut reagieren zu können. So wird implizit Wissen über zukünftige Entwicklungen in die Lösungsfindung integriert. Der Algorithmus kann

aus diesem Grund der Kategorie PFA zugeordnet werden (vgl. Unterabschnitt 2.1.5). Abbildung 4.46 visualisiert beispielhaft verschiedenen Lösungen von Planungsalgorithmen, erzeugt von den bereits vorgestellten Ansätzen, die nahe optimale Rundreisen planen. Die gelöste dynamische Probleminstanz ist aus der statischen Instanz CMT04, eingeführt in Christofides (1976) mit einem DOD von 0,2 generiert.

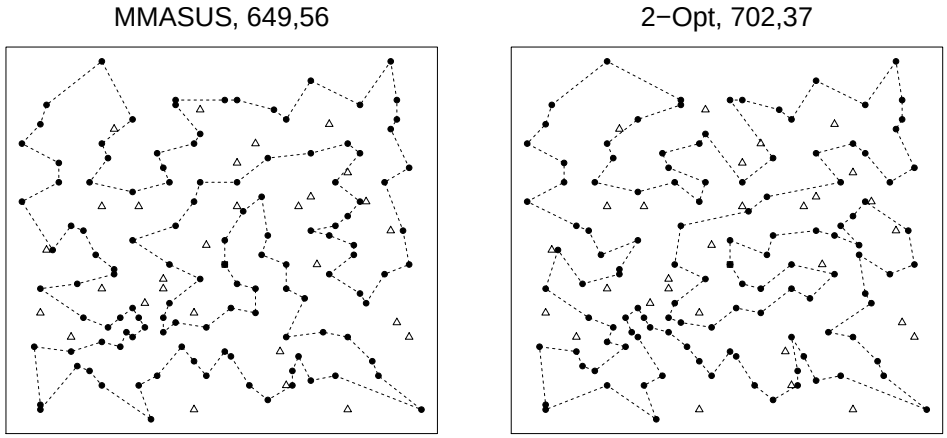

Abbildung 4.46: Verschiedene initiale Lösungen mit Lösungskosten

Bei der genauen Analyse der geplanten Rundreisen ist festzustellen, dass die Algorithmen dazu tendieren im äußeren Bereich der Probleminstanz Kundenanfragen zuerst zu bedienen. Zusätzlich wird versucht alle Kunden in einem Bereich, als Nachbarn in einer Rundreise zu planen. So wird verhindert, dass ein bereits besuchtes Gebiet einer Probleminstanz nochmals besucht wird. Dadurch sollen die Lösungskosten minimiert werden. Im Kontext eines DVRP kann sich dieser Sachverhalt aber auch als Nachteil erweisen. Tritt eine dynamische Kundenanfrage in einem Gebiet auf, in dem schon alle statischen Kundenanfragen bedient worden sind, werden die zusätzlichen Kosten, verursacht durch das Einplanen des dynamischen Kunden in die bestehende Rundreise, voraussichtlich groß. Führt hingegen eine initial geplante Route mehrfach durch ein Gebiet der Probleminstanz, ist die Wahrscheinlichkeit hoch, dass die zusätzlichen Kosten, verursacht durch das Einplanen eines dynamischen Kunden in die bestehende Rundreise, eher klein sind. Der Grund dafür liegt darin, dass die neu bekannt werdende dynamische Anfrage sich wahrscheinlich in der Nähe von noch nicht bedienten, bereits eingeplanten Anfragen befindet. Abbildung 4.47 visualisiert diesen Sachverhalt anhand einer Beispielprobleminstanz, die mit den zwei verschiedenen Ansätzen gelöst wird. Im linken Bild der Abbildung wird eine nahe optimale Lösung für die Probleminstanz geplant. Alle statischen Kundenanfragen im linken Bereich der Instanz werden als Nachbarn in eine Rundreise integriert. Der dynamische Kunde DK wird bekannt, wenn das Fahrzeug bereits den statischen Kunden E bedient. Die zusätzlichen Routingkosten, um den dynamischen Kunden in die bestehende Route einzuplanen, ergeben sich aus $c_{r_1} = 2 * a$. Im rechten Bild wird eine flexible Rundreise geplant. Sowohl der linke als auch der rechte Bereich der Probleminstanz werden durch das Fahrzeug zweimal zu

unterschiedlichen Zeiten besucht. Wenn der dynamische Kunde DK bekannt wird, bedient das Fahrzeug gerade den statischen Kunden B. Die zusätzlichen Routingkosten, um DK in die bestehende Route zu integrieren, ergeben sich aus $c_{r_2} = b + c - |\vec{BC}|$. Die zusätzlichen Kosten entstehend durch den dynamischen Kunden DK sind in der nahe optimalen Route größer als in der flexiblen Route ($c_{r_1} > c_{r_2}$).

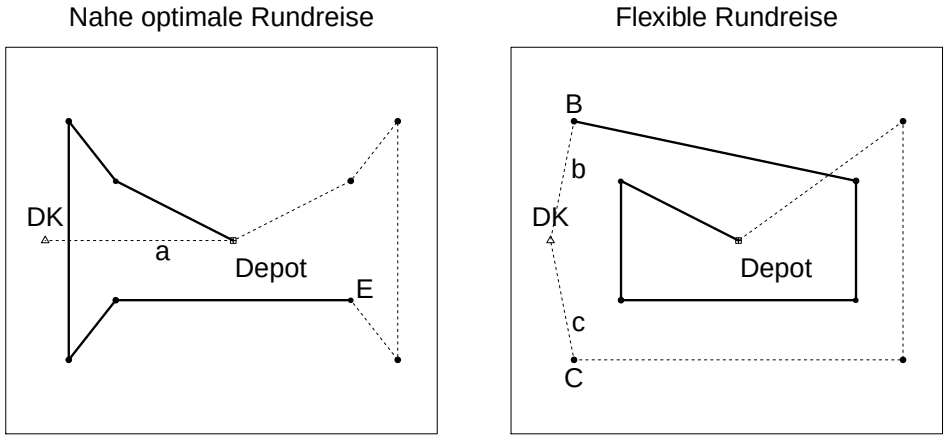

Abbildung 4.47: Vergleich der zusätzlichen Lösungskosten in einer nahe optimalen Rundreise (linkes Bild) und in einer flexiblen Rundreise (rechtes Bild)

Obwohl die zusätzlichen Kosten, entstehend durch das Einplanen des Kunden DK, in der flexiblen Rundreise kleiner sind, sind die Gesamtkosten der flexiblen Rundreise größer im Vergleich zur nahe optimalen Reise. Die Vorteile der flexiblen Reise werden sich erst auf die Gesamtkosten auswirken, wenn eine entsprechende Anzahl an dynamischen Kunden verstreut an den Rändern des Problemraums in der Problemstellung formuliert werden. Ein Algorithmus, der flexible Rundreisen plant, wird sich bei einem mittleren DOD (vgl. Abschnitt 2.1.3) und, zum Beispiel, bei einem großen LDOD oder einem großen Wert für die *FE* (vgl. Abschnitt 4.1.3) bewähren. Die aufgestellte Vermutung wird in der These 4.2.1 zusammengefasst. Die formulierte These 4.2.1 wird in dem Abschnitt 5.2 aufgegriffen und mithilfe von Experimenten bestätigt.

These 4.2.1. *Die Lösungsqualität einer flexibel geplanten Rundreise ist in Probleminstanzen mit mittlerem DOD und einem hohen Werte für die Flächen-Eigenschaft FE hoch.*

Um eine möglichst flexible Rundreise zu planen, müssen die unterschiedlichen Bereiche des Problemraums in einer Planung mehrfach und zu verschiedenen Zeiten berücksichtigt werden. Eine Möglichkeit der Umsetzung ist hier eine spiralförmige Abarbeitung der zu bedienenden Kunden. Das linke Bild in Abbildung 4.48 visualisiert die Grundidee des implementierten Algorithmus. Der Problemraum wird dabei in Zellen aufgeteilt, die Route wird jetzt spiralförmig von innen nach außen, orientiert an den Rasterzellen geplant (vgl. Abbildung 4.48). So wird gewährleistet, dass ein Bereich des Problemraums mehrfach durch die Route befahren wird. Zusätzlich werden Kunden im Zentrum des

Problemraums zeitig bedient. Das verhindert hohe Integrationskosten von dynamischen Anfragen weit entfernt vom Zentrum des Problemraums.

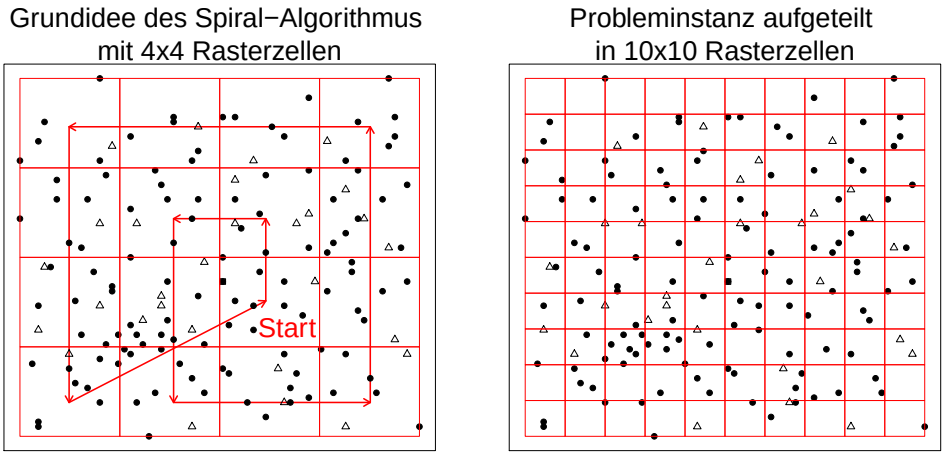

Abbildung 4.48: Grundidee des spiralförmigen Bedienens der Kundenaufträge mit einem Raster der Größe 4×4 und 10×10

Wie häufig ein Bereich des Problemraums berücksichtigt wird, kann durch die Anzahl der Rasterzellen definiert werden. Das rechte Bild in Abbildung 4.48 zeigt ein in 10×10 Zellen aufgelöstes Raster. Im Vergleich zu einer geringeren Auflösung des Rasters (vgl. linkes Bild in Abbildung 4.48), werden hier die Bereiche des Problemraums häufiger zu unterschiedlichen Zeiten besucht. Ein hoch aufgelöstes Raster vergrößert die Flexibilität, erhöht aber auch die entstehenden initialen Kosten. Zur Ermittlung einer geeigneten Rastergröße werden verschiedene, zufällig generierte, geclusterte und ungeclusterte Probleminstanzen erzeugt. Insgesamt basiert die Untersuchung auf rund 6.000 verschiedenen Probleminstanzen mit unterschiedlichen Werten für den *Degree of Dynamism* (DOD) (vgl. Abschnitt 2.1.3). Die Probleminstanzen werden mit dem Spiralalgorithmus mit den Rastergrößen $r \times r$, wobei $r \in \{4, 6, 8, ..., 16\}$ ist, gelöst. Als Anpassungsalgorithmus für die Experimente wird der bereits eingeführte Insertion-Algorithmus verwendet. Abbildung 4.49 zeigt die Ergebnisse der durchgeführten Experimente.

Die linke Spalte der Abbildung zeigt die Ergebnisse der Rasteranalyse für nicht geclusterte Probleminstanzen und die rechte Spalte für geclusterte. Auf der x-Achse sind die zwischen 0 und 1 normierten Routingergebnisse abgetragen. Kleine Werte entsprechen guten Ergebnissen. In den Zeilen der Abbildung 4.49 werden die maximale Anzahl, der Median und die durchschnittliche Anzahl an Kunden in einer Rasterzelle betrachtet. Die Anzahl der Kunden ist in Prozent zu der Gesamtanzahl aller Kunden angegeben. Gut ist zu sehen, dass die Punktwolken teilweise eine halbmondförmige Struktur aufweisen. Interessante Bereiche für die Bestimmung der Rastergröße sind Gebiete, bei denen viele gute Lösungen erzeugt werden und möglichst wenige schlechte. Die Gebiete entsprechen den Mittelstücken der Halbmonde und sind in Abbildung 4.49 mit roten Markierungen hervorgehoben. Überschneiden sich die Gebiete auf der y-Achse für geclusterte und ungeclusterte Probleminstanzen können die Werte für die Anzahl der

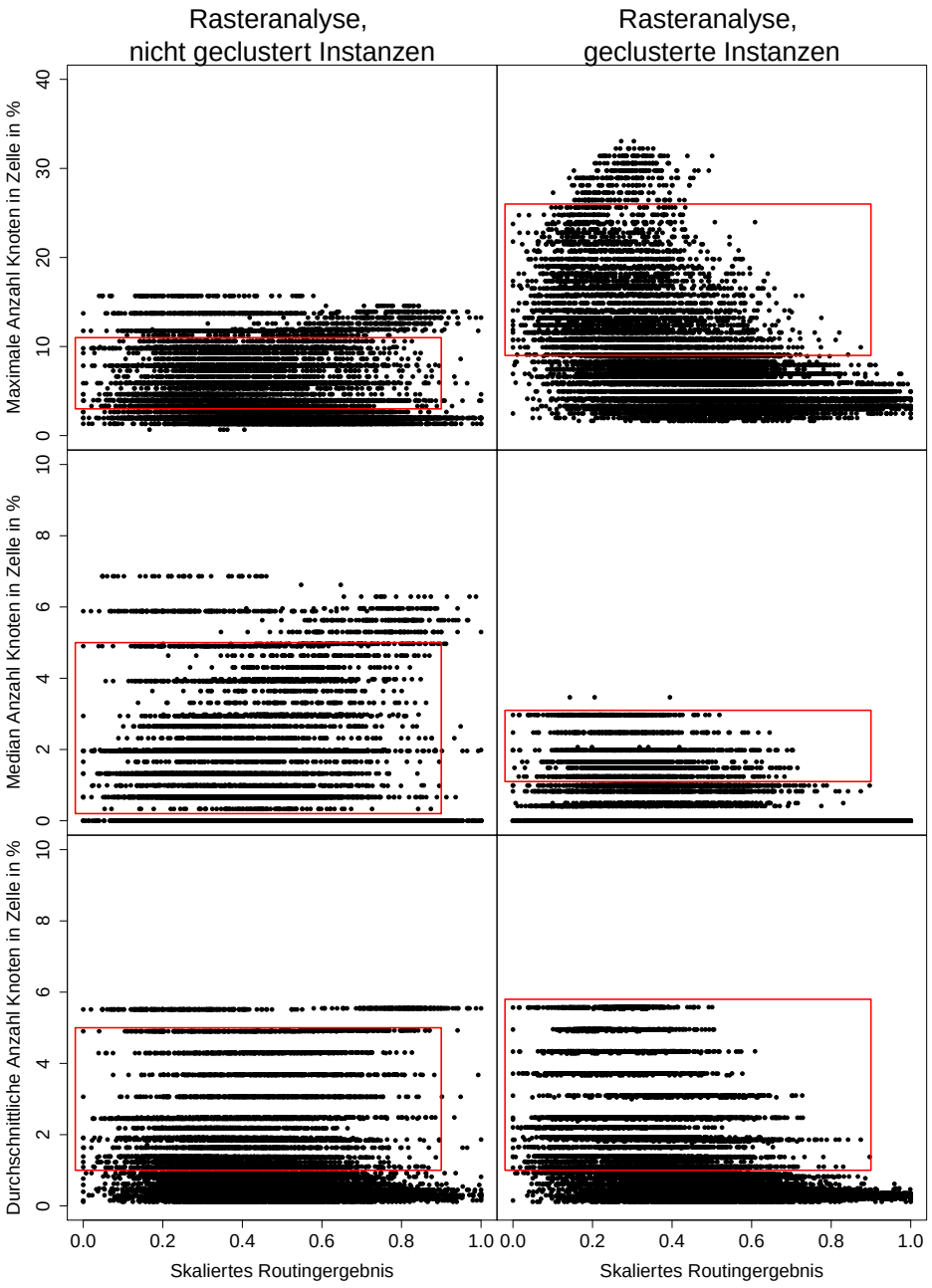

Abbildung 4.49: Beziehungen zwischen Routingergebnis und maximaler (erste Zeile), median (zweite Zeile) und durchschnittlicher (dritte Zeile) Anzahl an statischen Kunden in einer Rasterzelle für nicht geclusterte Instanzen (linke Spalte) und geclusterte Instanzen (rechte Spalte) (rote Markierungen zeigen für die Evaluation interessante Bereiche)

Kunden in einer Rasterzelle für die Bestimmung der Rastergröße herangezogen werden. So ergibt sich, dass maximal 9 bis 11 %, bzw. 1,5 bis 3 % im Median, und 1 bis 5 % im Durchschnitt der statischen Kunden in einer Rasterzelle sein dürfen, um viele gute und wenige schlechte Routingergebnisse zu erzielen. Der implementierte Algorithmus zur Bestimmung der Rastergröße mithilfe der maximalen Anzahl an Kunden in einer

Rasterzelle wird nun kurz erläutert.

1. Initialisierung der Rastergröße $r \times r$ mit $r = 3$
2. Ermittlung der maximalen Anzahl an Kunden in einer Rasterzelle max_K in Prozent
3. Wenn $max_K > 10\%$, erhöhe r um eins ($r = r + 1$) und fahre mit Schritt 2 fort, wenn $max_K \leq 10\%$, gib r als Rastergröße zurück

Zu beachten ist, dass eine geeignete Rastergröße auf der Grundlage von zufällig erzeugten Probleminstanzen bestimmt wird. Zufällig erzeugte Instanzen weisen Ausprägungen von Eigenschaften auf, die annähernd einer Normalverteilung folgen (vgl. Abschnitt 4.3). Über die Eignung der ermittelten Rastergröße für Probleminstanzen, die an den Rändern der Normalverteilung zu finden sind, lässt sich demzufolge keine Aussage ableiten. Aus diesem Grund bietet der Algorithmus, neben der dynamischen Ermittlung der Rastergröße, auch die Möglichkeit, eine feste Größe vorab zu konfigurieren. So können, zum Beispiel, Erfahrungswerte oder Ergebnisse weiterer Experimente durch den Algorithmus berücksichtigt werden.

Nach der Ermittlung der Rastergröße wird der erste statische Kunde v_1 der Route bestimmt. Der erste zu bedienende Kunde ist der mit der geringsten Entfernung zum Problemraummittelpunkt. Alle statischen Kunden in der zu v_1 gehörigen Rasterzelle werden mithilfe der Nächsten-Nachbar (NN)-Heuristik (vgl. Unterabschnitt 4.2.1) in die Route integriert. Die Rasterzellen werden jetzt spiralförmig bearbeitet, indem die Kunden in den Zellen mit der NN-Heuristik der Route hinzugefügt werden. Nachdem alle statischen Kunden in die Route eingeplant sind, wird diese mithilfe des Insert-Operators (vgl. Unterabschnitt 4.2.2) verbessert. Dabei wird für eine Auswahl an statischen Kunden geprüft, ob eine andere Position in der Route die Gesamtkosten reduziert. Wenn das der Fall ist, wird der statische Kunde aus der aktuellen Position entfernt und an der neuen Position eingefügt. Um die spiralförmige Struktur der geplanten Route zu erhalten, werden nur 15 % der Kunden auf eine mögliche Verbesserung hin überprüft. Die zu überprüfenden Kunden werden gleichmäßig verteilt aus der Route ausgewählt. Der Spiralalgorithmus ist bei r Rasterzellen, m Knoten in einer Rasterzelle und n Knoten in $O(r * m^2 + 0{,}15 * n^2)$. Die nachfolgende Aufzählung fasst das Vorgehen des Spiralalgorithmus zusammen. Abbildung 4.50 visualisiert eine spiralförmige Route als Planungsergebnis des Spiralalgorithmus.

1. Wenn keine feste Rastergröße konfiguriert ist, dann bestimme die Rastergröße mithilfe des vorgestellten Algorithmus
2. Definition des ersten zu planenden Kunden
3. Bestimmung der zu bearbeitenden Zelle
4. Planung der Kunden in der Zelle nach NN-Heuristik

5. Wenn noch eine nicht bearbeitete Zelle vorhanden ist, wähle die nächste zu bearbeitende Zelle und gehe zu Schritt 4, gibt es keine zu bearbeitende Zelle mehr, gehe zu Schritt 6

6. Wähle 15 % der geplanten Kunden gleichmäßig verteilt aus der Route und prüfe hinsichtlich einer Kostenverbesserung

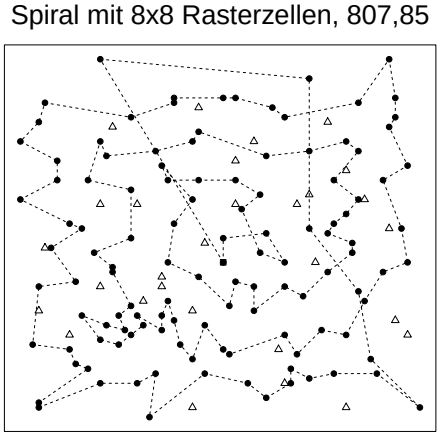

Abbildung 4.50: Spiralförmigen Route mit Lösungskosten als Planungsergebnis des Spiralalgorithmus

Der Spiralalgorithmus ist mit der Zielstellung entwickelt worden, Probleminstanzen mit bestimmten Ausprägungen von Eigenschaften mit besonders hoher Qualität zu lösen. Diese Zielstellung wird mit der in dieser Arbeit beschriebene Version erfolgreich umgesetzt (vgl. Kapitel 5). Der entwickelte Algorithmus soll zudem Ausgangspunkt für Weiterentwicklungen sein, die in zukünftigen Arbeiten untersucht und umgesetzt werden können.

4.2.8 Betrachtete Variationen der Lösungsansätze

Der erste betrachtete Planungsalgorithmus **NN 2-Opt** ist eine Kombination aus der NN-Heuristik und dem 2-Opt-Operator und ist aus diesem Grund der Kategorie Re-Optimierung (RO) zuzuordnen. Der lokale Optimierungsalgorithmus 2-Opt wird auf die Lösung, konstruiert durch die NN-Heuristik, angewendet. Der 2-Opt-Operator führt solange lokale Optimierungen durch, bis keine Verbesserung der Lösung mehr erreicht werden kann. Der Planungsalgorithmus **NN 2-Opt** wird mit den Anpassungsalgorithmen **Insert**, **US** und **NN 2-Opt** kombiniert. Der Insert-Operator führt lediglich eine minimale partielle Planrevision durch. Der dynamische Knoten wird gierig in die bestehende Tour zwischen zwei benachbarte Knoten eingeplant. Der US-Operator fügt den dynamischen Kunden nicht zwangsläufig zwischen zwei benachbarte Knoten in die bestehende Tour ein, führt also eine partielle Planrevision durch. Der als Anpassungsalgorithmus implementierte US-Operator berücksichtigt beide Stringing-Operatoren und

führt zusätzlich eine lokale Optimierung mit der Kombination von Unstringing- und Stringing-Operatoren durch. Der Anpassungsalgorithmus **NN 2-Opt** führt eine totale Planrevision durch. Eine komplett neue Route mit allen noch disponiblen Knoten wird ausgehend vom aktuellen Knoten mithilfe der NN-Heuristik geplant. Diese Route wird durch den 2-Opt-Operator verbessert. Der **NN 2-Opt** Anpassungsalgorithmus löst hier ein *Open VRP* (OVRP). Um sicherzustellen, dass die geplante Route valide ist, also im Depot endet, werden die realen Kosten zwischen Ausgangspunkt der Route (Standort des Fahrzeugs zum Zeitpunkt der Neuplanung) und Depot für die Planung mit 0 angenommen. So plant der Algorithmus mit großer Wahrscheinlichkeit den Ausgangspunkt der Route und das Depot als Nachbarn in die Rundreise. Der Anpassungsalgorithmus **NN 2-Opt** wird so häufig ausgeführt, bis eine valide Lösung als Ergebnis zur Verfügung steht. Die Auswahl verschiedener Startknoten führt zu unterschiedlichen Lösungen.

Der zweite betrachtete Planungsalgorithmus **MMAS US** kombiniert den MMAS Algorithmus mit dem lokalen Optimierungsoperator US und ist demzufolge ebenfalls der Kategorie Re-Optimierung (RO) zuzuordnen. **MMAS US** wird mit den Anpassungsalgorithmen **Insert** und **US** kombiniert. Eine Implementierung von **MMAS US** als Anpassungsalgorithmus ist im Zuge der Arbeit getätigt worden. Diese wird aber in den durchgeführten Experimenten aufgrund der schlechten Laufzeit des Algorithmus vernachlässigt.

Der dritte Planungsalgorithmus **Spiral** gehört der Kategorie *Policy Function Approximation* (PFA) an. Dieser plant also keine nahe optimale Rundreise, sondern versucht eine möglichst robuste und damit flexible Route zu kreieren, um auf mögliche dynamische Kundenanfragen gut reagieren zu können. Der **Spiral**algorithmus wird ausschließlich mit dem Anpassungsalgorithmus **Insert** kombiniert. Eine partielle bzw. total Planrevision des Anpassungsalgorithmus hätte die Zerstörung der Spiralstruktur zur Folge. Der **Spiral**algorithmus ermittelt die Rastergröße dynamisch (vgl. Unterabschnitt 4.2.7). Das Raster kann aber auch initial konfiguriert werden. Vereinzelte Experimente zeigen, dass eine Rastergröße von 8×8 äußerst robuste Ergebnisse für Probleminstanzen mit stark ausgeprägter Flächen-Eigenschaft und LDOD liefert. Um weitere Erkenntnisse hinsichtlich der fest konfigurierten Rastergröße zu sammeln, wird zusätzlicher der Planungsalgorithmus **Spiral8x8** betrachtet. Dieser passt die Rastergröße nicht dynamisch an, sondern verwendet eine feste Größe von 8×8. Der **Spiral8x8**-Algorithmus wird ebenfalls ausschließlich mit dem Anpassungsalgorithmus **Insert** kombiniert.

Insgesamt werden in der vorliegenden Arbeit sieben Algorithmen berücksichtigt. Die folgende Auflistung fasst alle betrachteten Kombinationen von Planungs- und Anpassungsalgorithmen zusammen.

- NN 2-Opt/Insert
- NN 2-Opt/US
- NN 2-Opt/NN 2-Opt
- MMAS US/Insert

- MMAS US/US
- Spiral/Insert
- Spiral8x8/Insert

4.3 Zusammenfassung

Das Kapitel 4 führt die im Zuge dieser Arbeit entwickelten Problemeigenschaften für das DVRP ein. Mithilfe dieser Eigenschaften lassen sich erstmalig dynamische Problemstellungen voneinander unterscheiden und anhand von Gemeinsamkeiten oder Unterschieden gruppieren. Für einen fairen Vergleich von Algorithmen ist die Unterscheidbarkeit von Probleminstanzen Voraussetzung. Die vorliegende Arbeit bestätigt in Abschnitt 5.2, dass Algorithmen existieren, die nur für Problemstellungen mit bestimmten Ausprägungen von Eigenschaften erfolgreich sind (vgl. Unterabschnitt 4.2.7). Einen solchen Algorithmus für einen Vergleich auf der Basis von Problemstellungen heranzuziehen, die andere Ausprägungen von Eigenschaften aufweisen, ist nicht zielführend. Leicht kann der Eindruck entstehen, dass ein bestimmter Algorithmus einem anderen ganz allgemein überlegen ist. Das widerspricht aber den No-Free-Lunch-Theoremen, eingeführt von Macready und Wolpert (1996) (vgl. Abschnitt 2.2). Die Theoreme zeigen, dass diese Überlegenheit lediglich für Problemstellungen mit bestimmten Ausprägungen von Problemeigenschaften gilt. Es ist fundamental bei einem Vergleich von Algorithmen, die zugrundeliegenden Problemstellungen zu charakterisieren. Oftmals wird dieser Sachverhalt aber vernachlässigt. Auch bei der Verwendung von zufällig generierten Probleminstanzen müssen diese mithilfe ihrer Eigenschaften beschrieben werden. In Abschnitt 4.1 wurde gezeigt, dass die Ausprägungen der Eigenschaften bei der zufälligen Generierung annähernd einer Normalverteilung folgen (vgl. Abbildung 4.3). Algorithmen, die erfolgreich für Ausprägungen der Eigenschaften an den Rändern der Normalverteilung sind, werden so systematisch benachteiligt. Mithilfe der in dieser Arbeit entwickelten Eigenschaften wird eine zielgerichtete Analyse realer Problemstellungen ermöglicht. So kann evaluiert werden, ob reale Instanzen existieren, die Ausprägungen der Eigenschaften an den Rändern der Normalverteilung vorweisen und wie häufig diese sind. Es kann der Frage nachgegangen werden, ob zufällig erzeugte Probleminstanzen eine gute Approximation der Realität sind. Die in dieser Arbeit eingeführten Problemeigenschaften bilden die Voraussetzung für eine erfolgreiche Selektion von Algorithmen und erlauben zusätzlich eine zielgerichtete Generierung von Probleminstanzen für die Analyse oder Konstruktion von Lösungsansätzen. Für die Evaluation der Eigenschaften, unabhängig von Algorithmenselektion, werden Probleminstanzen unterschiedlichen Typs generiert. Es wird gezeigt, dass die Typen der Instanzen mithilfe der Problemeigenschaften eindeutig identifiziert werden können. Zusätzlich werden Probleminstanzen erarbeitet, die eine minimale bzw. maximale Ausprägung der Problemeigenschaften aufweisen. Diese Visualisierungen sollen einen Planer dabei unterstützen die Ausprägung von Eigenschaften an Strukturen von dynamischen und statischen Kundenanfragen einzuschätzen und zu

vergleichen. Auf diese Weise kann die Ausprägung von wichtigen Problemeigenschaften in Probleminstanzen auch manuell nachvollzogen werden. So kann eine Selektion von Algorithmen, auch ohne Werkzeugunterstützung durch einen Planer stattfinden. Die Eigenschaften, die einen großen Einfluss auf die Lösungsqualität von Algorithmen haben werden im folgenden Kapitel 5 erarbeitet.

In Abschnitt 4.2 werden alle betrachteten Algorithmen im Detail besprochen. Berücksichtigt werden verschiedene Variationen von Lösungsansätzen. Unter den betrachteten Algorithmen ist der Spiral/Insert-Algorithmus, der im Zuge dieser Arbeit entwickelt wird. Der Planungsalgorithmus konstruiert eine Spiralfahrt, der Anpassungsalgorithmus fügt dynamische Anfragen mit einem einfachen Insert-Operator (vgl. Unterabschnitt 4.2.2) der Spirale hinzu. Es wird theoretisch gezeigt, dass sich eine flexible Rundreise, realisiert durch eine Spiralfahrt, für Problemstellungen mit bestimmten Ausprägungen von Problemeigenschaften positiv auf die Gesamtlösungskosten des Algorithmus auswirken. Es wird also nachgewiesen, dass die in dieser Arbeit eingeführten Problemeigenschaften Ausgangspunkt für die gezielte Konstruktion von Algorithmen sein können. Der Erfolg des Spiral/Insert-Algorithmus wird im nachfolgenden Kapitel 5 mithilfe von Experimenten nachgewiesen.

Die erarbeiteten Problemeigenschaften und Lösungsansätze sind die Voraussetzung für die Exploration des Problemraums (vgl. Abbildung 2.9 in Unterabschnitt 2.2.1). Auf der Basis der Lösungsqualitäten der Algorithmen werden im nachfolgenden Kapitel 5 Probleminstanzen zielgerichtet generiert. Diese Instanzen bilden die Voraussetzung für die Analyse von den Beziehungen zwischen Problemeigenschaften und Lösungsqualität und sind die Basis für die Konstruktion von Modellen für die erfolgreiche Selektion von Algorithmen für Online-Optimierungsprobleme am Beispiel des DVRP.

Kapitel 5

Exploration und Analyse von Problemen, Lösungen und Eigenschaften

Nachdem im vorherigen Kapitel neue Problemeigenschaften für das DVRP eingeführt und betrachtete Algorithmen vorgestellt wurden, sollen im nun folgenden Kapitel die Beziehungen zwischen den von Algorithmen generierten Problemlösungen und den Eigenschaften erläutert werden. Ein Ziel der Betrachtungen ist die Identifikation von Problemeigenschaften, die sich wesentlich auf die Problemschwere für Algorithmen auswirken. Ein weiteres Ziel ist die Analyse, ob sich identifizierte Eigenschaften für die manuelle Auswahl von Algorithmen eignen. Ausgangspunkt für die Bearbeitung der Zielstellungen dieses Kapitels ist die Erzeugung einer Datenbasis mithilfe eines Data-Farming-Experiments (vgl. Unterabschnitt 2.2.1), die Problemstellungen mit den zugehörigen Lösungen enthält. Auf der Grundlage dieser Datenbasis, werden auch die im nachfolgenden Kapitel 6 diskutierten Modelle bzw. Optimierer, für die automatisierte Selektion von Algorithmen, erzeugt. Abbildung 5.1 visualisiert den in dieser Arbeit umgesetzten Lösungsansatz. Die Komponenten Eigenschaften, Lösungen und Probleminstanzgenerator, mit dem sich das nachfolgende Kapitel im Detail beschäftigt, sind grau hervorgehoben.

Abschnitt 5.1 führt den im Zuge dieser Arbeit umgesetzten Probleminstanzgenerator ein. Zusätzlich wird das umfangreiche Data-Farming-Experiment beschrieben, mit dem sowohl schwere als auch leichte Probleminstanzen für alle betrachteten Algorithmen erzeugt werden. Diese Ergebnisse der Experimente bilden die Datenbasis für alle weiteren Untersuchungen. Bei der Umsetzung des Experiments werden erstmalig wesentliche Eigenschaften von Algorithmen für DVRP erarbeitet, die bei einer Generierung von repräsentativen Lösungen für Problemstellungen berücksichtigt werden müssen. Die erzeugten und gelösten Instanzen werden in Unterabschnitt 5.1.3 bezüglich ihrer Weiterverwendbarkeit analysiert. Dabei wird evaluiert, ob repräsentative Instanzen für die betrachteten Algorithmen durch den Probleminstanzgenerator gefunden werden und ob die verwendeten Algorithmen hinreichend komplementär zueinander sind. Auf der

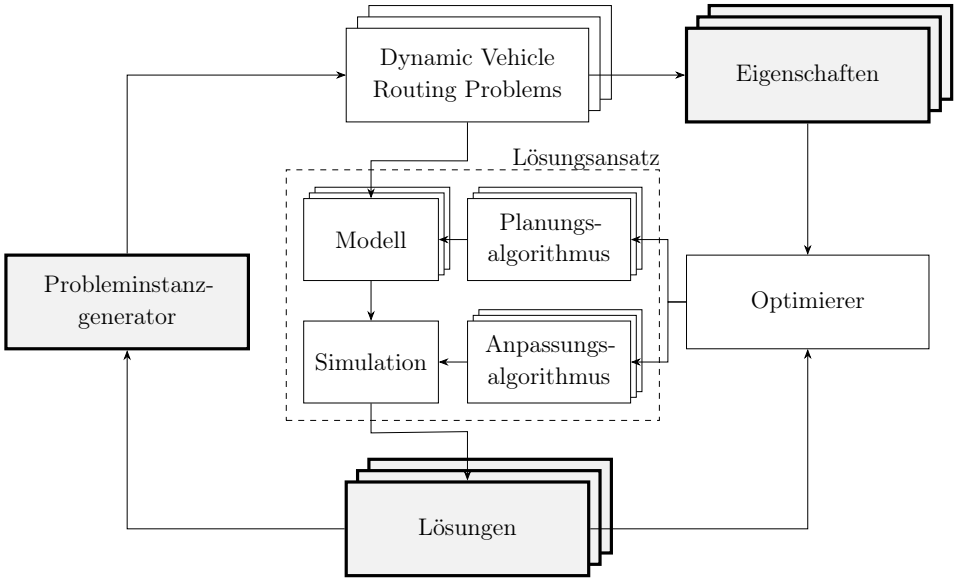

Abbildung 5.1: Lösungsansatz mit Fokus auf Probleminstanzgenerator sowie Eigenschaften und Lösungen

Grundlage des erfolgreichen Nachweises der Eignung der erzeugten Probleminstanzen, kann in Abschnitt 5.2 die Analyse der Beziehungen zwischen den von Algorithmen generierten Problemlösungen und den Eigenschaften durchgeführt werden. Ein Schwerpunkt der Analyse ist die Identifikation von einflussreichen Problemeigenschaften für die Unterteilung in leichte bzw. schwere Probleminstanzen. Ein weiterer Schwerpunkt ist die Ermittlung der Wertebereiche der Eigenschaften, um auch eine manuelle Auswahl von Algorithmen, neben der Umsetzung des Optimierers (vgl. Abbildung 5.1), zu ermöglichen. Das Kapitel diskutiert die Zusammenhänge zwischen Problemschwere und Problemeigenschaften und führt den ersten Nachweis, dass eine Selektion von Algorithmen auf der Grundlage der in dieser Arbeit eingeführten Problemeigenschaften möglich ist. Das Kapitel schließt mit einer Zusammenfassung des Erarbeiteten.

5.1 Erzeugung und Evaluation von Instanzen und Lösungen

Für die Evaluation der Schwere von DVRP und die erfolgreiche Selektion von Algorithmen auf der Basis von Problemeigenschaften, ist die Menge von Performances Y (vgl. Abbildung 2.9), die sich für diese Arbeit aus den Lösungen der Probleminstanzen ergeben, essenziell. Die gezielte Generierung von Instanzen mithilfe von Data-Farming-Experimenten ist im Kontext von Algorithmenselektion eine etablierte Vorgehensweise, um für Algorithmen repräsentative Probleminstanzen zu erzeugen (vgl. Unterabschnitt 2.2.3). Repräsentative Instanzen sind Problemstellungen, bei denen Algorithmen besonders gute bzw. schlechte Ergebnisse erzielen. Im Nachfolgenden wird der Probleminstanzgenerator, der im Zuge dieser Arbeit entwickelt worden ist, vorgestellt.

Im Anschluss werden die Experimente erläutert, mit denen eine Vielzahl von Probleminstanzen und Lösungen erzeugt werden. In Unterabschnitt 5.1.3 werden abschließend die Ergebnisse der Experimente hinsichtlich ihrer Aussagekraft und Weiterverwendbarkeit untersucht.

5.1.1 Der Probleminstanzgenerator

Der Probleminstanzgenerator hat die Aufgabe aus einer Vielzahl von möglichen DVRP-Instanzen diejenigen zu finden, für die ein Algorithmus besonders gute bzw. schlechte Ergebnisse erzielt. Für die Realisierung der Suche, wird, wie in Abschnitt 4.1.1, auf den Genetischer Algorithmus (GA) auf Basis des Meta-Heuristik-Frameworks SEREIN, eingeführt in Uhlig (2015), zurückgegriffen. Die Aufgabe des Algorithmus besteht darin, eine gegebene statische VRP-Instanz in eine dynamische zu transformieren. Ziel der Transformation ist die Erzeugung einer Probleminstanz, für die die Lösungskosten, erzeugt durch einen Algorithmus, minimal bzw. maximal ausgeprägt sind. Die Transformation beschränkt sich dabei auf die Auswahl von dynamischen Kunden aus der Menge der gegebenen Kunden. Die Anzahl wird durch den DOD definiert. Der implementierte Algorithmus besteht aus vier Hauptkomponenten, die im Folgenden kurz erläutert werden. Die Komponenten sind ähnlich wie in Abschnitt 4.1.1 umgesetzt.

- **GA-Modell der Probleminstanz** - Die DVRP-Instanz wird mithilfe einer Liste von Integerwerten repräsentiert. Die Werte entsprechen den Identifikatoren der Kunden. Mithilfe der Identifikatoren kann ein Bezug zur X-/Y-Koordinate der Kunden hergestellt werden. Ein Modell entspricht einem Individuum.

- **Mutationsoperator** - Der Operator vertauscht Werte im GA-Modell. Es werden ausschließlich dynamische mit statischen Kunden vertauscht. Die ersten $n_{dynamisch}$ Kunden im GA-Modell repräsentieren dynamische Kunden.

- **Rekombinationsoperator** - Der Operator kombiniert zwei GA-Modelle. Alle dynamischen Kunden, die in beiden Modellen vorkommen, werden in das neue Modell übernommen. Die verbleibende Anzahl dynamischer Kunden werden zufällig aus den dynamischen Kunden der Modelle ausgewählt.

- **Fitnessfunktion** - Die Funktion übersetzt das GA-Modell in eine DVRP-Instanz. Die Probleminstanz wird in ein Simulationsmodell überführt und mithilfe des RVRPSim evaluiert. Grundlage für die Evaluation durch den Simulator ist der gegebene Lösungsansatz, bestehend aus Planungs- und Anpassungsalgorithmus. Das verarbeitete Ergebnis der Simulation entspricht der Fitness des GA-Modells.

Basierend auf einer statischen VRP-Instanz und einem DOD erzeugt der Generator eine Startpopulation bestehend aus x Individuen. Jedes Individuum der Population wird mithilfe der Fitnessfunktion evaluiert. Der Lösungsansatz, für den Probleminstanzen gefunden werden sollen, wird der Funktion vorgegeben. Die Ergebnisse der Evaluation

bilden die Grundlage für die Generierung einer Folgepopulation aus den Individuen mithilfe der Mutations- bzw. Rekombinationsoperatoren. Der Probleminstanzgenerator speichert alle erzeugten Individuen mit den Ergebnissen der Evaluation in einer Datenbank. Die gespeicherten Daten bilden die Grundlage für die Evaluation der Schwere von DVRP und für die Modellbildung für die erfolgreiche Selektion von Algorithmen. Grundsätzlich gilt, dass eine heuristische Suche nicht zuverlässig ein Minimum bzw. Maximum finden kann, sondern sich diesen optimal konstruierten Probleminstanzen nur annähert. Ergebnisse nahe dem realen Minimum bzw. Maximum sind hier aber durchaus ausreichend, weil das Verfahren zuverlässig schwer bzw. leicht zu lösende Probleminstanzen findet. Es ist nicht notwendig die schwerste bzw. die leichteste Instanz zu identifizieren, weil davon ausgegangen werden kann, dass diese Instanzen ähnlich zu schwer bzw. leicht zu lösenden sind.

5.1.2 Das Data-Farming-Experiment

Aus der eben beschriebenen Funktionsweise des Probleminstanzgenerators ergeben sich die in Tabelle 5.1 gelisteten Eingabeparameter für das Data-Farming-Experiment (vgl. Unterabschnitt 2.2.3). Neben den Parametern zeigt Tabelle 5.1 auch die möglichen Ausprägungen, die im Zuge dieser Arbeit untersucht werden.

Tabelle 5.1: Übersicht über die Ausprägungen der Eingabeparameter der Experimente

Eingabeparameter	**Ausprägungen**
Algorithmus	NN 2-Opt/Insert; NN 2-Opt/US; NN 2-Opt/NN 2-Opt; MMAS US/Insert; MMAS US/US; Spiral8x8/Insert; Spiral/Insert (vgl. Unterabschnitt 4.2.8)
Suchrichtung	Minimierung; Maximierung
Statische Probleminstanz	CMT01; CMT02; CMT11; CMT12 von Christofides (1976); CMT11TM50R; CMT12TM50R erarbeitet im Zuge dieser Arbeit; Golden_01; Golden_05; Golden_13; Golden_17 von Golden et al. (1998)
DOD	0,2; 0,3; 0,5

Abgesehen von dem Algorithmus und der Suchrichtung muss eine statische Probleminstanz für ein Experiment definiert werden. Die statische Instanz ist Ausgangspunkt für die Erzeugung dynamischer Probleminstanzen. In der Literatur werden eine Vielzahl von Benchmarkproblemstellungen für das VRP diskutiert. Häufige Instanzen beschreiben gleichmäßig verteilte oder in Clustern angeordnete Kunden. Die durchgeführten Experimente basieren auf den gleichmäßig verteilten Probleminstanzen CMT01 und CMT02 eingeführt von Christofides (1976). In CMT01 werden 50 und in CMT02 75 Kunden, von denen Anfragen ausgehen, modelliert. In den Instanzen CMT11 und

CMT12 sind die Kunden in Clustern angeordnet, wobei in CMT11 große Cluster mit 120 und in CMT12 kleine Cluster mit 100 Kunden modelliert werden. Probleminstanzen, in denen eine Kombination aus gleichmäßig verteilten und in Clustern angeordneten Kunden vorkommen, sind die Instanzen CMT11TM50R und CMT12TM50R, die im Zuge dieser Arbeit erstellt werden. Für die Generierung werden 50 zufällige Anfragen aus den Instanzen CMT11 bzw. CMT12 ausgewählt und zufällig in der Problemfläche verteilt. Statische Problemstellungen, eingeführt von Golden et al. (1998), bei denen die Kunden in künstlichen Mustern angeordnet sind, vervollständigen die ausgewählten Instanzen. Als Muster werden ineinander angeordnete Kreise (Golden_01) und Sterne (Golden_17), ein mit Kunden ausgefülltes Quadrat (Golden_13) und ein ausgefüllter Kreis (Golden_05) betrachtet. In den Probleminstanzen Golden_01, Golden_17, und Golden_05 werden 120 und in Golden_13 96 Kunden modelliert. Alle betrachteten statischen Probleminstanzen sind in Abbildung 5.2 abgebildet.

Die ausgewählten statischen Probleminstanzen repräsentieren eine Vielzahl von verschiedenen Typen. So wird sichergestellt, dass die Suche nach guten bzw. schlechten Instanzen für einen Algorithmus erfolgreich ist. Zudem wird gewährleistet, dass auch schlechte bzw. gute Probleminstanzen mit unterschiedlichen Merkmalsausprägungen gefunden werden können. Die Daten für das Training der Optimierer für die Auswahl von Algorithmen ergeben sich aus den in Tabelle 5.1 abgebildeten Probleminstanzen. Die verwendete Vielfalt garantiert, dass Optimierer ganz allgemeine Zusammenhänge zwischen Problemeigenschaften und Lösungsqualität aus der erzeugten Datenbasis ableiten können. So sollen die Optimierer in der Lage sein, mit dem abgeleiteten Wissen, auch für unbekannte Probleminstanzen erfolgreich Algorithmen auszuwählen. Ein entsprechender Nachweis wird in Kapitel 6 geführt. Der begrenzende Faktor für die betrachtete Vielfalt von Probleminstanzen ist die Laufzeit eines Experiments, auf die im weiteren Verlauf des Abschnitts genauer eingegangen wird.

Jede mögliche Kombination von Ausprägungen der Parameter in Tabelle 5.1 ergibt ein Experiment. Um die Anzahl der durchgeführten Experimente zu beschränken, werden für den DOD ausschließlich Werte aus der Menge $M = \{0{,}2; 0{,}3; 0{,}5\}$ betrachtet. Die Auswahl der Werte ermöglicht die Untersuchung, ob Optimierer auch für dynamische Problemstellungen mit unbekannter Dynamik erfolgreich Algorithmen auswählen können. In Kapitel 6 wird unter anderem gezeigt, dass ein Optimierer auf der Grundlage von gelernten Zusammenhängen zwischen Eigenschaften und Lösungsqualität von Problemstellungen mit einem DOD $\in M$ auch für Probleminstanzen mit einem DOD von 0,4 erfolgreich Algorithmen auswählt. Insgesamt werden 420 Experimente für die Generierung von repräsentativen Probleminstanzen durchgeführt. Pro Experiment werden 20 Individuen in 50 Generationen durch den GA entwickelt. Insgesamt werden also 420.000 Probleminstanzen erzeugt, die durch den RVRPSim evaluiert werden müssen. Alle Ergebnisse werden in einer Datenbank gespeichert und bilden die Datenbasis für die weiteren Analysen.

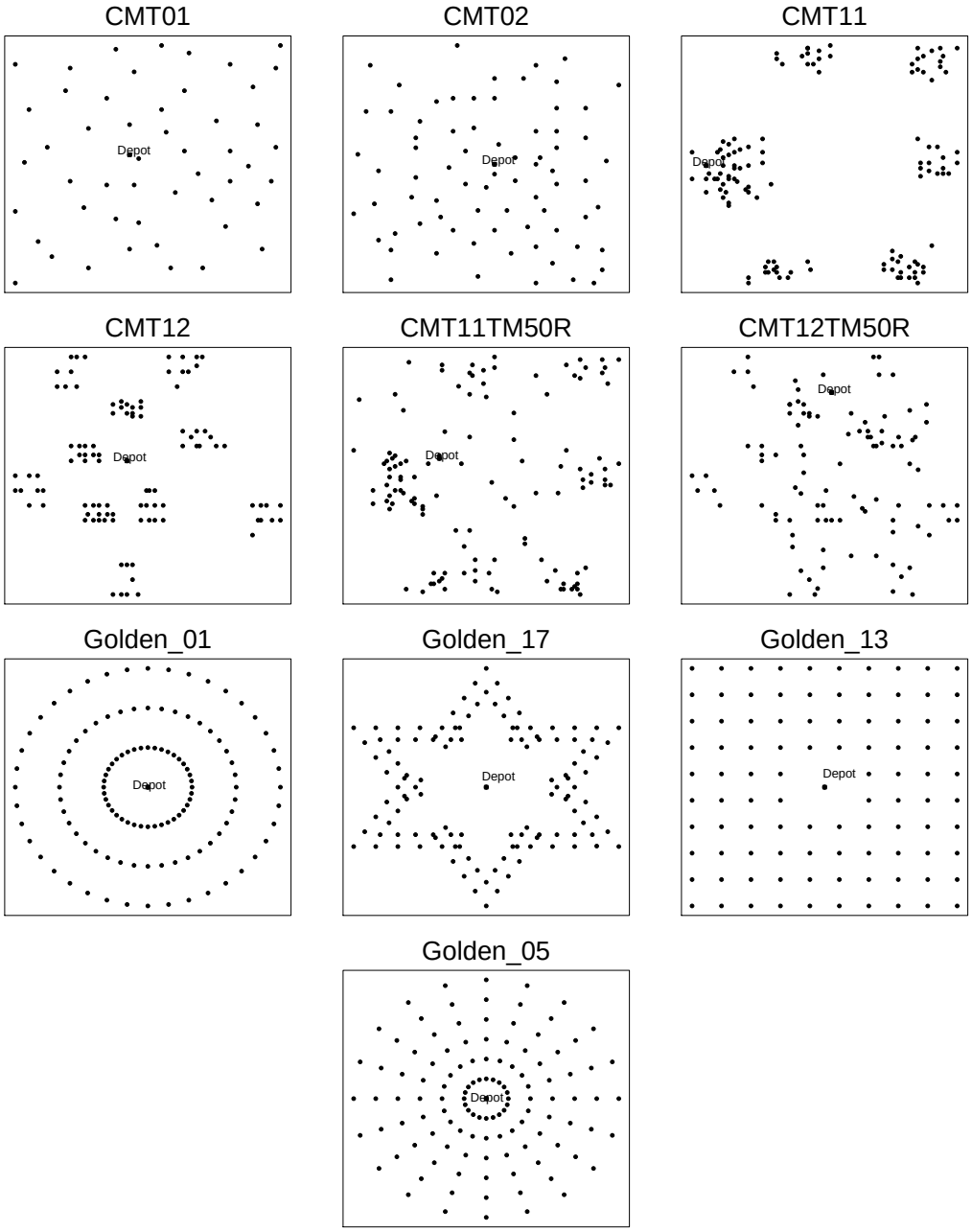

Abbildung 5.2: Für das Data-Farming-Experiment verwendete statische Probleminstanzen

Um eine repräsentative Evaluation der Probleminstanzen zu gewährleisten, müssen Besonderheiten bei dynamischen Problemstellungen berücksichtigt werden. Abbildung 5.3 zeigt eine DVRP-Instanz zum Zeitpunkt $t_0 = 0$. Die statischen Kunden sind in eine Route geplant, der dynamische Kunde DK ist noch nicht bekannt. Die zum Zeitpunkt t_0 kalkulierten Kosten sind unabhängig von der Fahrrichtung. Sowohl der Kunde A als auch B können zuerst bedient werden, bei gleichen zu erwartenden Kosten.

Beide Graphen in Abbildung 5.4 zeigen den Zustand der DVRP-Instanz aus Ab-

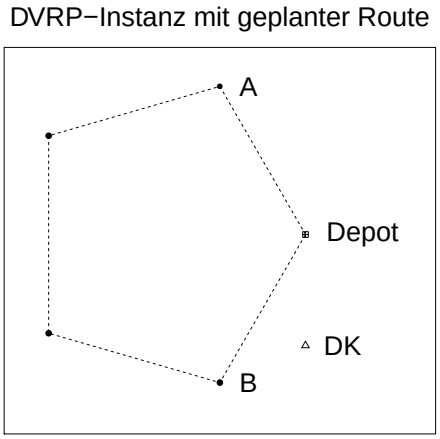

Abbildung 5.3: DVRP-Instanz mit einer geplanten Route (bei gleichen Kosten können die Kunden A oder B zuerst bedient werden)

bildung 5.3 zum Zeitpunkt $t_1 > t_0$, an dem der dynamische Kunde DK bekannt wird. Im linken Graph wird der Kunde A zuerst bedient. Die zusätzlichen Kosten, erzeugt durch das Einplanen von DK, sind gering. Im rechten Graph wird der Kunde B zuerst bedient. Hier sind die zusätzlichen Kosten signifikant größer. Die zu erwartenden Kosten eines Lösungsansatzes werden also maßgeblich von der Startrichtung der Fahrt beeinflusst. Um eine Evaluation einer Probleminstanz unabhängig von der Startrichtung zu gewährleisten, muss eine erzeugte Lösung für beide Startrichtungen evaluiert werden.

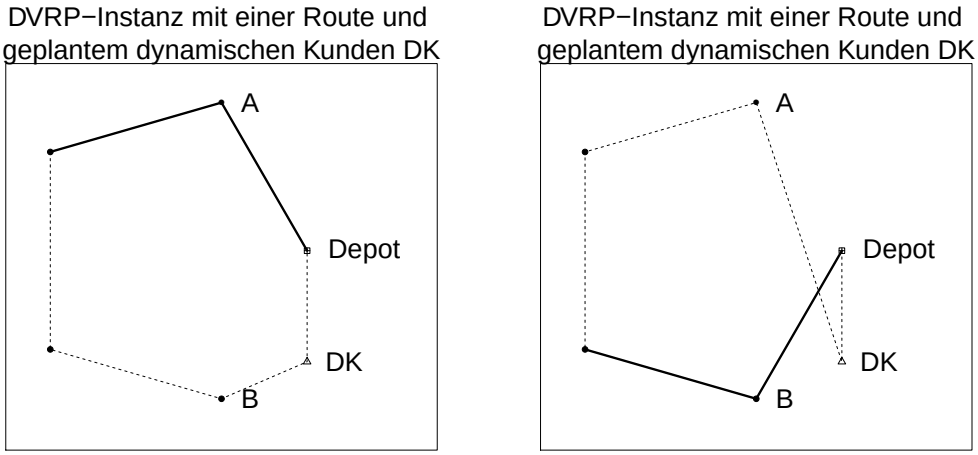

Abbildung 5.4: Auswirkung der Startrichtung auf die Mehrkosten verursacht durch das Einplanen eines dynamischen Kunden (DK)

Eine weitere Besonderheit, die es bei der Evaluation der Probleminstanzen zu berücksichtigen gilt, ist der Zeitpunkt, an dem dynamische Kunden bekannt werden. Für die Experimente werden die Ankunftszeiten der dynamischen Kunden äquidistant über die Zeitlinie bis zum Zeithorizont T verteilt. Die Generierung der Zeitpunkte t_i erfolgt nach der folgenden Regel: $t_{i+1} - t_i = d_i; d_i = d_{i+1} \forall i$, wobei $t_{n+1} = T$. T ergibt sich aus der Lösung der zugrundeliegenden statischen Problemstellung. Die Zuteilung

von dynamischen Kunden auf den konkreten Zeitpunkt des Bekanntwerdens erfolgt zufällig. Abbildung 5.5 zeigt eine DVRP-Instanz zum Zeitpunkt $t_0 = 0$ mit den beiden dynamischen Kunden DK1 und DK2.

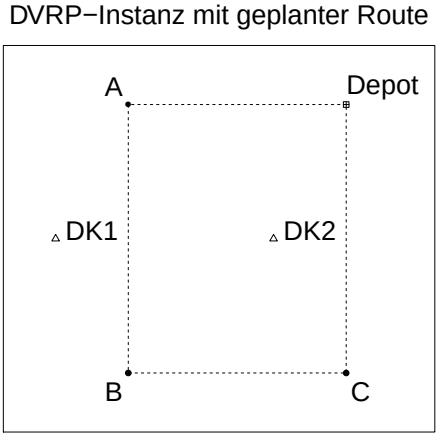

Abbildung 5.5: Symmetrische DVRP-Instanz mit den dynamischen Kundenanfragen DK1 und DK2

Beide Graphen in Abbildung 5.6 zeigen die DVRP-Instanz aus Abbildung 5.5 zum Zeitpunkt $t_1 > t_0$. Der linke Graph in der Abbildung zeigt das Szenario, in dem der Kunde DK1 zum Zeitpunkt t_1 bekannt wird. Im rechten Graphen der Abbildung wird der Kunde DK2 zum Zeitpunkt t_1 bekannt. In beiden Szenarien können die dynamischen Kunden mit ähnlichen Mehrkosten in die bestehende Route integriert werden.

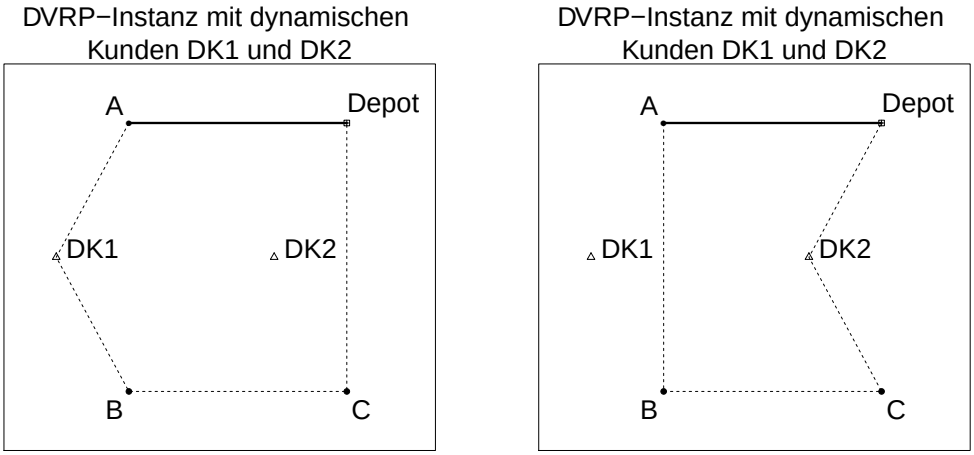

Abbildung 5.6: Probleminstanz aus Abbildung 5.5 zum Zeitpunkt $t_1 > 0$ (im linken Graph erscheint DK1 zu t_1, im rechten Graph erscheint DK2 zu t_1)

Die beiden Graphen in Abbildung 5.7 zeigen den Zustand der Szenarien aus Abbildung 5.6 zum Zeitpunkt $t_2 > t_1$. Im rechten Graphen wird der Kunde DK2 und im linken Graphen der Kunde DK1 zum Zeitpunkt t_2 bekannt. Gut ist zu sehen, dass aufgrund der fortgeschrittenen Zeit und der zeitlichen Reihenfolge des Bekanntwerdens

der dynamischen Kunden, das Szenario, repräsentiert im linken Graphen, zu günstigeren Gesamtkosten abgearbeitet werden kann, als das Szenario im rechten Graphen. Die Struktur der Probleminstanzen ist identisch, ausschließlich die Ankunftszeiten der dynamischen Kunden unterscheiden sich.

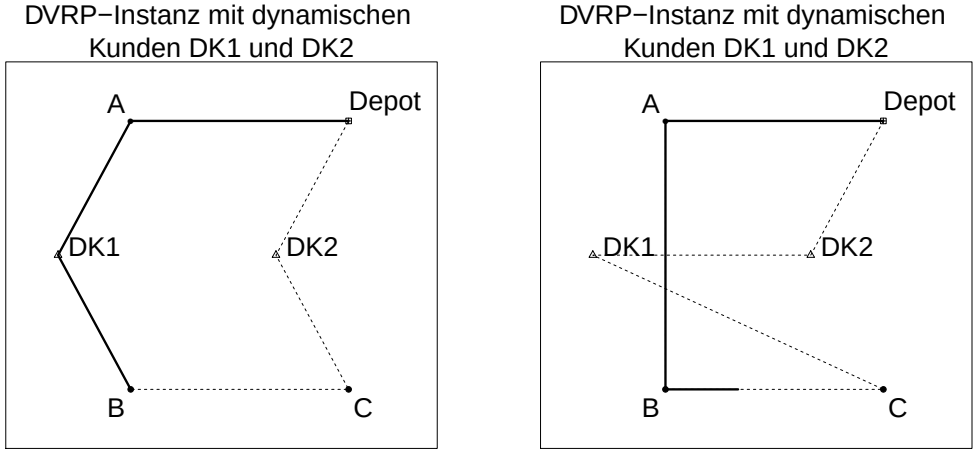

Abbildung 5.7: Probleminstanz aus Abbildung 5.5 zum Zeitpunkt $t_2 > t_1$ (im linken Graph erscheint DK2 zu t_2, im rechten Graph erscheint DK1 zu t_2)

Neben der Startrichtung hat also auch die Zuordnung von den dynamischen Anfragen auf die konkreten Zeitpunkte einen erheblichen Einfluss auf die Lösungskosten für eine DVRP-Instanz. Dieser Aspekt muss ebenfalls bei der Evaluation der Probleminstanzen berücksichtigt werden. Da nicht alle verschiedenen Zuordnungen von den Anfragen auf die Zeitpunkte evaluiert werden können, werden pro Startrichtung 20 zufällig gewählte Zuordnungen betrachtet. Pro erzeugter Instanz und Lösung werden demzufolge 40 Simulationen durchgeführt. Aufgrund der stochastischen Eigenschaften der Lösungsalgorithmen, werden zusätzlich pro Instanz 10 verschiedene Lösungen erzeugt. Das Durchschnittsergebnis aller 400 Simulationen ist das Endergebnis der Evaluation einer Probleminstanz.

Für die Evaluation der 420.000 Probleminstanzen werden demzufolge 168 Millionen Simulationen benötigt. Zu beachten ist, dass auf der Grundlage der eben beschriebenen Experimente noch kein Vergleich der Algorithmen untereinander möglich ist. Für keine der Instanzen stehen Lösungen von mehr als einem Algorithmus zur Verfügung. Aus diesem Grund werden die Probleminstanzen, erzeugt für die Algorithmen Spiral/Insert und NN 2-Opt/US (vgl. Unterabschnitt 4.2.8), zusätzlich mit den verbleibenden Algorithmen gelöst. Dabei müssen weitere 288 Millionen Simulationen durchgeführt werden. Die beschriebenen Experimente werden auf einem Computercluster mit 16 Kernen ausgeführt. Die Berechnungszeit betrug rund sechs Wochen. Dabei wurden >50 Gigabyte (GB) an Rohdaten für die spätere Analyse erzeugt.

5.1.3 Evaluation und Analyse der Experimente

Im folgenden Abschnitt werden die Ergebnisse der Data-Farming-Experimente kurz evaluiert. Schwerpunkt der Untersuchung ist, ob repräsentative Instanzen für die Algorithmen durch die Suche gefunden werden und ob die verwendeten Algorithmen komplementär zueinander sind. Es wird erörtert, ob Instanzen existieren, für die ein Algorithmus besonders gute bzw. schlechte Ergebnisse erzielt und ob für jeden Algorithmus Probleminstanzen existieren, die dieser, im Vergleich zu den anderen, am besten löst.

Abbildung 5.8 visualisiert die Verteilung der Ergebnisse des Algorithmus NN 2-Opt/Insert für dynamische Probleminstanzen, generiert aus der statischen Instanz CMT01 mit einem DOD von 0.2. Die Ergebnisse sind in Form des erweiterten *Value of Information* erfasst (vgl. Unterabschnitt 2.1.6).

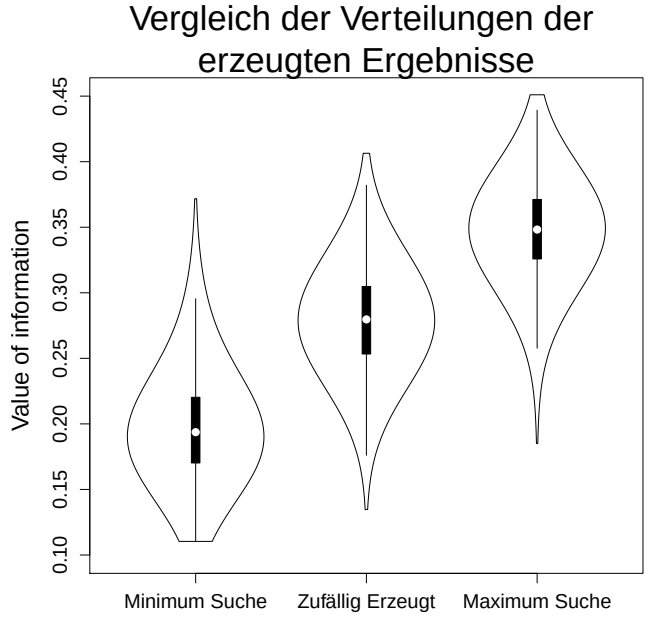

Abbildung 5.8: Vergleich der Verteilungen der erzeugten Ergebnisse

Die Verteilungen, dargestellt in Abbildung 5.8, sind repräsentativ für alle betrachteten Algorithmen, Probleminstanzen und DOD. Das linke Box-Plot-Verteilungsdiagramm in Abbildung 5.8 zeigt die Ergebnisse bei der Suche nach Instanzen, für die der Algorithmus besonders gute Ergebnisse erzielt. Das rechte Box-Plot-Verteilungsdiagramm zeigt die Ergebnisse der Suche nach besonders schwer zu lösenden Probleminstanzen. Im mittleren Diagramm sind Ergebnisse für zufällig erzeugte Instanzen erfasst. Gut ist zu sehen, dass die gezielte Suche nach Probleminstanzen den gewünschten Effekt hat. Es werden viele besonders schwer bzw. leicht zu lösende Probleminstanzen generiert. Die durchschnittlichen Ergebnisse aus den Suchen sind signifikant voneinander entfernt. Im Vergleich zu der zufälligen Generierung, werden viele Probleminstanzen gefunden, die signifikant schwerer bzw. leichter für den Algorithmus zu lösen sind. Damit bilden die generierten Instanzen eine solide Grundlage für das Trainieren des Modells $S(f(x))$, das

auf der Grundlage von den Problemeigenschaften $f(x)$ einen Algorithmus α auswählt (vgl. Unterabschnitt 2.2.3). Der erweiterte *Value of Information* von 0,11 zeigt zudem, dass besonders gut zu lösende Probleminstanzen existieren. Grundsätzlich gilt aber zu bedenken, dass die Suche durch die gewählte Anzahl der Individuen und Generationen stark eingegrenzt ist (vgl. Unterabschnitt 5.1.2). Eine umfangreichere Suche führt voraussichtlich zu besseren Ergebnissen. Abbildung 5.8 zeigt, dass Probleminstanzen erzeugt werden, für die ein Algorithmus ganz unterschiedliche Lösungsqualitäten erreicht. Alle Instanzen haben aber einen DOD von 0,2. Die Ergebnisse rechtfertigen also zusätzlich die Einführung der Problemeigenschaften für die bessere Charakterisierung von Problemstellungen. Der Vergleich von Probleminstanzen und der von Algorithmen ausschließlich auf der Basis des DOD ist demzufolge unzureichend.

Abbildung 5.9 zeigt den Anteil für den ein Algorithmus das beste Ergebnis im Vergleich zu allen anderen Algorithmen erzielt. Die Datenbasis für die ermittelten Anteile stellen die Probleminstanzen dar, die bei der gezielten Suche nach Instanzen für den Spiral/Insert- und den NN 2-Opt/US-Algorithmus generiert und von allen Algorithmen gelöst sind (vgl. Unterabschnitt 5.1.2).

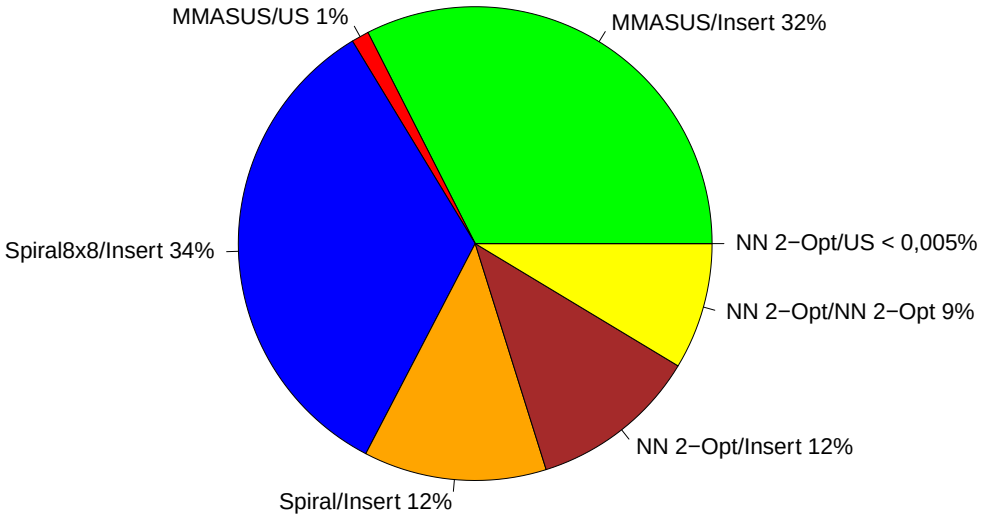

Abbildung 5.9: Anteil der Probleminstanzen, für den der zugeordnete Algorithmus das beste Ergebnis erzielt

Abbildung 5.9 zeigt, dass alle Algorithmen einen signifikanten Anteil an Probleminstanzen am besten lösen. Ausschließlich der NN 2-Opt/US-Algorithmus wird von einem oder mehreren anderen Algorithmen fast komplett überdeckt. Obwohl der zugrundeliegende Datensatz Instanzen enthält, die zielgerichtet für den Algorithmus generiert worden sind, ist der Anteil hier kleiner als 0,005%. Ein Algorithmus, dessen erzeugte Lösungen überwiegend schlechter sind als die eines anderen Algorithmus, wird

im Folgenden als überdeckter Algorithmus bezeichnet. Das Diagramm, abgebildet in Abbildung 5.9, bestätigt die Wichtigkeit der Charakterisierung der Problemstellungen beim Vergleich von Algorithmen. Für einen Vergleich aller betrachteten Algorithmen können Instanzen so ausgewählt werden, dass jeder betrachtete Algorithmus derjenige mit der höchsten Lösungsqualität ist. Werden, zum Beispiel, nur Probleminstanzen in den Vergleich aufgenommen, für die der Spiral8x8/Insert-Algorithmus der erfolgreichste ist, kann leicht der Eindruck entstehen, dass der Spiral8x8/Insert-Algorithmus den anderen Algorithmen generell überlegen ist. Ausschließlich die korrekte Charakterisierung der verwendeten Probleminstanzen kann den Erfolg oder den Misserfolg der zu vergleichenden Algorithmen richtig einordnen. Erst die im Kontext dieser Arbeit eingeführten Problemeigenschaften für das DVRP ermöglichen den fairen Vergleich von Algorithmen (vgl. Kapitel 4).

Abbildung 5.10 zeigt den direkten Vergleich der Algorithmen NN 2-Opt/US und NN 2-Opt/Insert. Der NN 2-Opt/Insert-Algorithmus liefert zu fast 100% bessere Ergebnisse im Vergleich zu NN 2-Opt/US, obwohl der US-Operator aufwendiger zu berechnen ist (vgl. Abschnitt 4.2). Der NN 2-Opt/US-Algorithmus wird demzufolge von NN 2-Opt/Insert überdeckt. Bei einer Selektion zwischen nur diesen beiden Algorithmen sollte man sich immer für den NN 2-Opt/Insert-Algorithmus entscheiden. Den Verdacht, dass der Insert- dem US-Operator grundsätzlich überlegen ist, bestätigt der direkte Vergleich zwischen den Algorithmen MMASUS/Insert und MMASUS/US nicht. Hier löst der MMASUS/US noch 4% der Probleminstanzen besser als der MMASUS/Insert-Algorithmus. Der Prozentuale Anteil erscheint niedrig. Zu beachten ist aber, dass der zugrundeliegende Datensatz keine Probleminstanzen enthält, die zielgerichtet für den MMASUS/US-Algorithmus konstruiert worden sind.

Abbildung 5.11 zeigt das Ergebnis der hierarchischen Clusteranalyse der Algorithmen nach Ward Jr (1963). Als Distanzmaß für das Clustering wird die euklidische Distanz zwischen den Ergebnissen der Algorithmen verwendet. Die Analyse identifiziert vier Cluster. Der Spiral/Insert-Algorithmus weist die größte Distanz zu den restlichen Algorithmen auf. Der Spiral8x8/Insert-Algorithmus ist ebenfalls in ein separates Cluster eingeordnet. Beide Algorithmen beziehen implizit Probleminformationen in den Lösungsprozess ein (vgl. Unterabschnitt 4.2.7) und unterscheiden sich voraussichtlich aus diesem Grund am deutlichsten von den anderen. Die verbleibenden Algorithmen werden in zwei weitere Cluster eingeteilt. Interessant ist hier, dass die Unterteilung nicht auf dem verwendeten Planungsalgorithmus basiert, sondern scheinbar auf dem Anpassungsalgorithmus. Die Algorithmen MMASUS/US und NN 2-Opt/US erzeugen ähnliche Ergebnisse und werden gemeinsam in ein Cluster eingeordnet. Die Algorithmen MMASUS/Insert und NN 2-Opt/Insert sind sich insgesamt am ähnlichsten und werden gemeinsam mit dem NN 2-Opt/NN 2-Opt-Algorithmus in ein Cluster einsortiert. Der Unterschied der Ergebnisse bei einer totalen (NN 2-Opt/NN 2-Opt) und einer minimalen partiellen Planrevision (MMASUS/Insert und NN 2-Opt/Insert) scheint hier also vergleichsweise gering zu sein. Zu bedenken ist, dass die Algorithmenlaufzeit bei einer totalen Revision erheblich höher ist. Zusammenfassend werden vier sehr unterschiedliche Klassen von Algorithmen durch

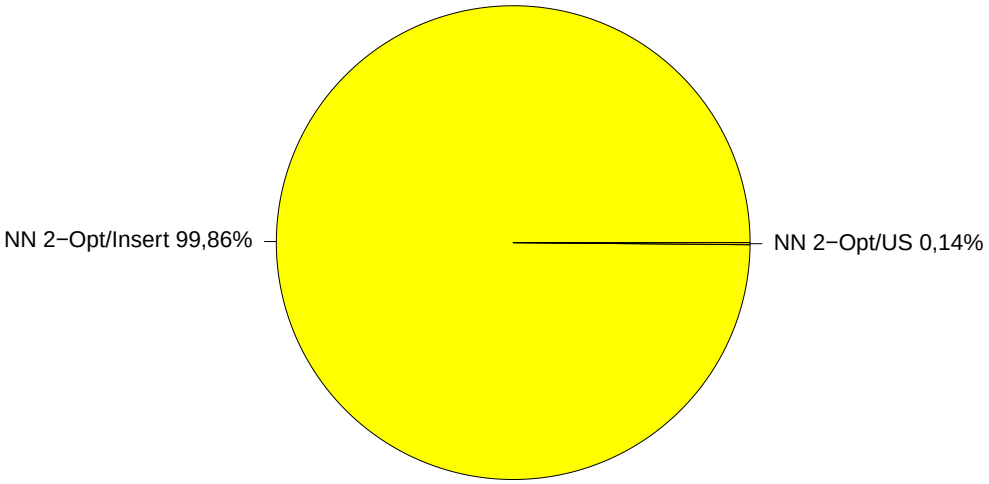

Abbildung 5.10: Paarweise Vergleich der Anteile an besten Ergebnissen von NN 2-Opt/Insert und NN 2-Opt/US

die Analyse identifiziert, wobei das Cluster mit den Algorithmen MMASUS/US und NN 2-Opt/US scheinbar von den Algorithmen aus den anderen Clustern dominiert wird (vgl. Abbildung 5.9). Die Evaluation der Ergebnisse der Data-Farming-Experimente zeigt, dass für alle Algorithmen repräsentative Instanzen existieren, auf deren Basis ein Modell für die Selektion von Algorithmen erstellt werden kann. Zusätzlich zeigt die Analyse, dass jeder Algorithmus, außer NN 2-Opt/US, eine signifikante Menge von Probleminstanzen am besten löst und damit für eine Selektion infrage kommt.

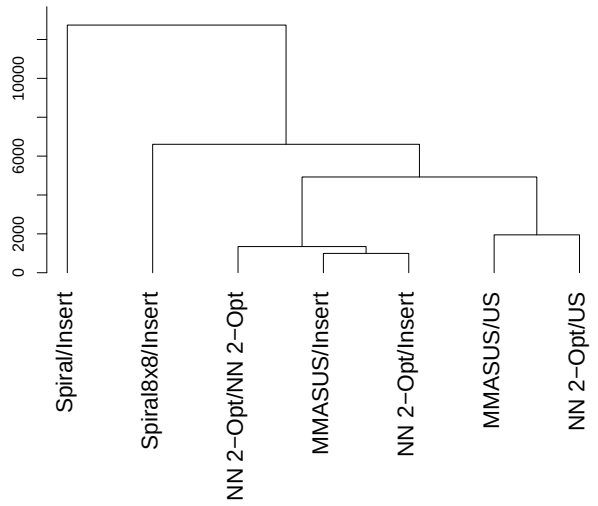

Abbildung 5.11: Hierarchischen Clusteranalyse der Algorithmen in einem Dendrogramm (die Tiefe des Diagramms repräsentiert den euklidischen Abstand)

5.2 Analyse der Beziehungen von Eigenschaften und Lösungen

Die in Abschnitt 5.1 generierten Probleminstanzen mit den zugehörigen Lösungen bilden zum einen die Grundlage für die Analyse der Schwere von DVRP und zum anderen für die Modellbildung $S(f(x))$ für die Selektion von Algorithmen. Die Instanzen werden durch die 181 in Abschnitt 4.1 eingeführten Problemeigenschaften repräsentiert. Für eine übersichtliche Erörterung der Problemschwere und auch für die Anwendung der Algorithmenselektion müssen die Problemeigenschaften hinsichtlich ihrer Wichtigkeit untersucht werden. Es ist nicht praktikabel für alle 181 Eigenschaften Prognosemodelle auf Basis von, zum Beispiel, historischen Daten für die Selektion von Algorithmen zu konstruieren. Unterabschnitt 5.2.1 identifiziert die Eigenschaften, die einen wesentlichen Einfluss auf die Performance der betrachteten Algorithmen haben. Um geeignete Algorithmen auszuwählen, müssen ausschließlich für die identifizierten wichtigen Eigenschaften Prognosemodelle über dynamische Ereignisse erstellt werden. Unterabschnitt 5.2.2 erläutert schwere und leichte Problemstellungen für die Algorithmen und analysiert zusätzlich das Verhalten der Algorithmen in Beziehung zu den wichtigen Problemeigenschaften.

5.2.1 Analyse des Einflusses der Eigenschaften

Den Einfluss bzw. die Wichtigkeit der Problemeigenschaften zu untersuchen, ist aus verschiedenen Gründen interessant. Eine Auswahl von Algorithmen auf Basis von Eigenschaften im Kontext einer dynamischen Problemstellung ist nur dann umsetzbar, wenn geeignete Prognosemodelle über die Problemeigenschaften existieren oder erarbeitet

werden können. Das Erstellen von 181 verschiedenen Modellen ist nicht praktikabel. Kann die Modellbildung aber auf wenige wichtige Eigenschaften reduziert werden, wird die Umsetzung der Algorithmenselektion auch im Kontext von dynamischen Problemstellungen sinnvoll realisierbar. Zusätzlich wird mit der Diskussion der einflussreichen Eigenschaften auch die Grundlage für die manuelle Auswahl von Algorithmen, basierend auf wenigen wichtigen Problemeigenschaften, gelegt. Neben der domänenspezifischen Notwendigkeit hat die Erarbeitung der wichtigen Problemeigenschaften auch ganz allgemein Vorteile. Guyon und Elisseeff (2003) weisen darauf hin, dass die Reduktion der Eigenschaften die Trainingsphase von Algorithmen zum maschinellen Lernen stark verkürzen kann. Zusätzlich wird die Komplexität der Modelle reduziert und damit die Interpretation der Ergebnisse vereinfacht. Durch die Eliminierung der Eigenschaften, die kaum zum Ergebnis beitragen, wird Rauschen verhindert und damit die Genauigkeit der Modelle erhöht. Häufig setzten sich Arbeiten, die sich mit der Selektion von Algorithmen beschäftigen, auch mit der Wichtigkeit von Problemeigenschaften auseinander, siehe, zum Beispiel, Wagner et al. (2018).

Eine Analyse des Einflusses der Eigenschaften kann, unter anderem, mithilfe einer Faktorenanalyse (Wirtz und Nachtigall, 2004) aber auch, zum Beispiel, auf der Grundlage von Ergebnissen, gesammelt durch Random-Forest-Modelle erfolgen. Random-Forest ist ein Klassifikations- und Regressionsverfahren etabliert von Breiman (2001). Das Verfahren basiert auf unkorrelierten, randomisierten Entscheidungsbäumen (vgl. Unterabschnitt 4.1.4). Jeder unabhängige Entscheidungsbaum in einem Random-Forest trifft eigenständige Entscheidungen. Das Random-Forest-Endergebnis ergibt sich aus den Ergebnissen aller Bäume. Für die Analyse des Einflusses der Eigenschaften wird für jeden betrachteten Algorithmus ein Random-Forest-Modell mit 2.000 binären Entscheidungsbäumen trainiert, das die Beziehungen zwischen Problemeigenschaften und Lösungsqualität modelliert (Regression). Beim Training der Bäume wird in jedem Knoten aus einer zufällig gewählten Teilmenge aller Eigenschaften (bei einer Konstruktion der Bäume nach Breiman, 2001) diejenige Eigenschaft ausgewählt, die die Daten am vielversprechendsten in zwei Teilmengen aufteilt. Für die Auswahl der Eigenschaft wird die Metrik *Gini impurity* herangezogen. Die *Gini impurity* ist eine rechnerisch effiziente Approximation der Entropie und misst, wie gut eine potenzielle Teilung der Daten in zwei Teilmengen auf der Grundlage der betrachteten Eigenschaft ist (Menze et al., 2009). Die *Gini importance* (vgl. Breiman, 2001) einer Eigenschaft berechnet sich aus der ermittelten *Gini impurity* der Eigenschaften an jedem Knoten in allen Bäumen und ist damit ein Maß dafür, wie häufig eine Eigenschaft für die Teilung der Daten herangezogen worden ist und wie groß der Wert der Eigenschaft für die erfolgreiche Konstruktion des Random-Forest-Modells ist (Menze et al., 2009). Abbildung 5.12 zeigt die acht einflussreichsten Problemeigenschaften basierend auf dem Durchschnitt der *Gini importance* aller Random-Forest-Modelle.

Die Eigenschaft, die am häufigsten für eine Teilung herangezogen wird, ist die Flächen-Eigenschaft *FE* (vgl. Abschnitt 4.1.3). Unter den fünf wichtigsten Eigenschaften befinden sich zusätzlich zwei Schwerpunkt-Eigenschaften *SE* und die Distanz-Eigenschaften

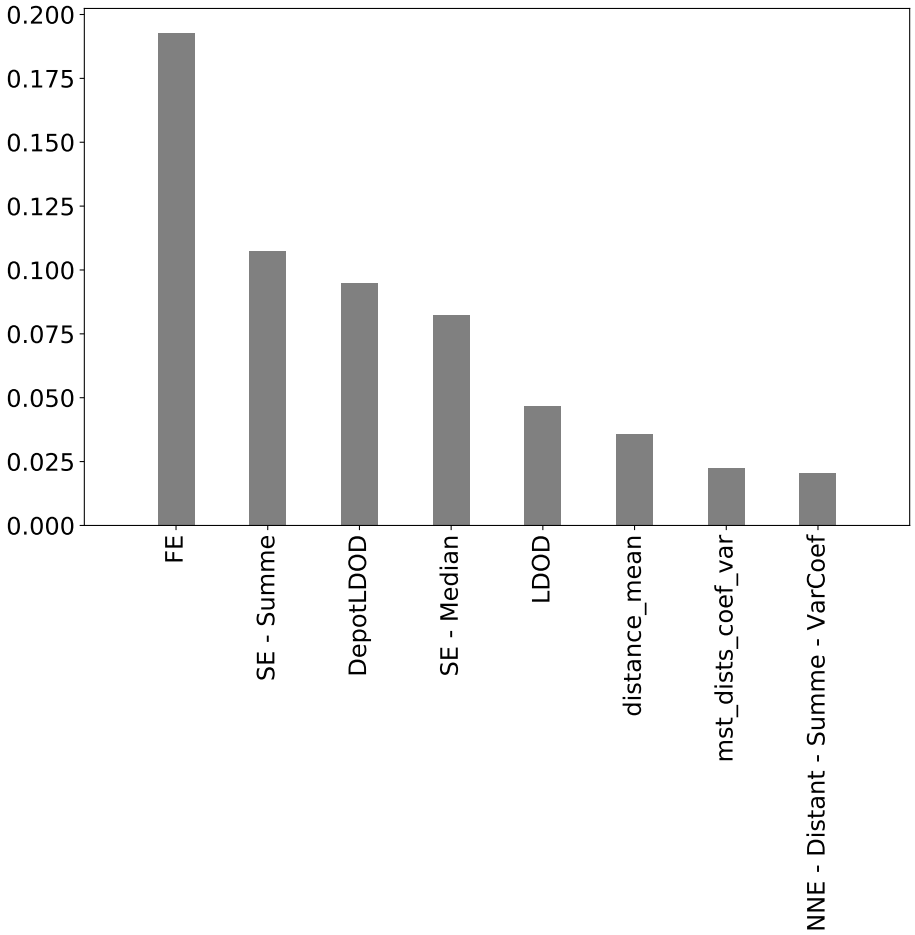

Abbildung 5.12: Die acht einflussreichsten Problemeigenschaften basierend auf dem Durchschnitt der Gini importance (Breiman, 2001) aller Random-Forest-Modelle

LDOD und DepotLDOD (vgl. Abschnitt 4.1.3). Interessant ist, dass der DOD nicht unter den wichtigsten Problemeigenschaften zu finden ist. Das bestätigt den Zusammenhang zwischen LDOD und DOD, formuliert vom Autor dieser Arbeit in Mayer et al. (2017). Der LDOD spiegelt neben der Lage der dynamischen Anfragen auch die Anzahl wider und überdeckt somit den DOD. Grundsätzlich gilt, dass für verschiedene Algorithmen unterschiedliche Eigenschaften wichtig sein können. Die in Abbildung 5.12 dargestellten Problemeigenschaften repräsentieren den durchschnittlichen Einfluss über allen betrachteten Algorithmen. Abbildung 5.12 bestätigt zudem die Notwendigkeit der in dieser Arbeit eingeführten Problemeigenschaften (vgl. Abschnitt 4.1). Sechs von den acht als wichtig für die Einschätzung der Performance von Algorithmen eingestuften Eigenschaften wurden im Zuge dieser Arbeit entwickelt.

5.2.2 Analyse schwerer und leichter Problemstellungen

Die in Unterabschnitt 5.2.1 erarbeiteten Eigenschaften, die einen großen Einfluss auf die Lösungsqualität der Algorithmen haben, bilden die Grundlage für die folgende Analyse von schweren und leichten Problemstellungen. Es soll untersucht werden, wie sich die Eigenschaften auf die Qualität der Lösungen der Algorithmen auswirken und ob es, basierend auf den Eigenschaften, ebenfalls möglich ist, eine Selektion von Algorithmen zu realisieren.

Für je einen Vertreter der vier in Abbildung 5.11 identifizierten Klassen von Lösungsalgorithmen werden diejenigen Probleminstanzen identifiziert, für die der Algorithmus die besten Lösungen generiert (vgl. Abbildung 5.9). Diese Probleminstanzen werden den Instanzen gegenübergestellt, für die der Algorithmus schlechtere Lösungen im Vergleich zu anderen Algorithmen erzeugt. Diese schlechteren Lösungen werden im Folgenden auch als überdeckte Lösungen bezeichnet. Für die Visualisierung der Beziehungen zwischen Lösungsqualität und Problemeigenschaften werden nur die jeweils besten bzw. schlechtesten 20 Prozent der Instanzen aus den Mengen herangezogen. Eine Untersuchung zeigt, wie sich die beiden Mengen von Probleminstanzen mithilfe der vier einflussreichsten Eigenschaften unterscheiden lassen. Zusätzlich werden Beispielinstanzen aus beiden Mengen gezeigt und auf der Grundlage der Erkenntnisse aus Abschnitt 4.1 diskutiert. Im Anschluss werden die Ähnlichkeiten bzw. Unterschiede der Algorithmen evaluiert und die Wertebereiche der einflussreichen Eigenschaften für die Probleminstanzen, für die die Algorithmen die besten Lösungen generieren, ermittelt. Abschließend wird diskutiert, ob mithilfe der Eigenschaften auch eine Algorithmenselektion möglich ist. In Abschnitt B.1 werden die in diesem Kapitel verwendeten Visualisierungen schwerer und leichter Problemstellungen mit zusätzlichen Interpretationshilfen abgebildet.

MMAS US/US

Abbildung 5.13 zeigt in den oberen beiden Diagrammen die Beziehungen zwischen den vier einflussreichsten Problemeigenschaften (FE, SE_{Summe}, SE_{Median}, $DepotLDOD$) und den Problemstellungen, für die der Algorithmus MMAS US/US die besten Lösungen (grün) bzw. schlechtere Lösungen im Vergleich zu anderen Algorithmen erzeugt (rot). Gut ist zu sehen, dass sowohl mithilfe der Eigenschaftenkombinationen FE - SE_{Summe} und SE_{Median} - $DepotLDOD$ die beiden Mengen gut voneinander zu unterscheiden sind. Der Algorithmus liefert Ergebnisse, die im Vergleich zu allen anderen Algorithmen besser sind, wenn die Werte für SE_{Summe} und $DepotLDOD$ klein sind. Grundsätzlich sind sich die beiden Eigenschaften ähnlich, wenn sich das Depot in der Nähe des geometrischen Schwerpunktes befindet. Für mittlere und große Werte der beiden Eigenschaften erzeugt der Algorithmus ausschließlich Lösungen, die von anderer Algorithmen überdeckt werden. Auf die Eigenschaften SE_{Median} und FE reagiert der Algorithmus robuster. Hier werden beste Lösungen für geringe und mittlere Werte der Eigenschaften generiert. Ausschließlich wenn die Werte für die Eigenschaften sehr groß werden ($FE > 0{,}8$; $SE_{Median} > 0{,}8$) können nur noch überdeckte Lösungen generiert werden. Die in Ab-

bildung 5.13 gezeigten Ergebnisse bestätigen die Analysen der geplanten Rundreisen der Algorithmen aus Unterabschnitt 4.2.7, die die Erarbeitung des Spiralalgorithmus motiviert haben. Auf die Beziehung zwischen den Ausprägungen der Eigenschaften und dem Spiralalgorithmus wird im Folgenden noch genauer eingegangen.

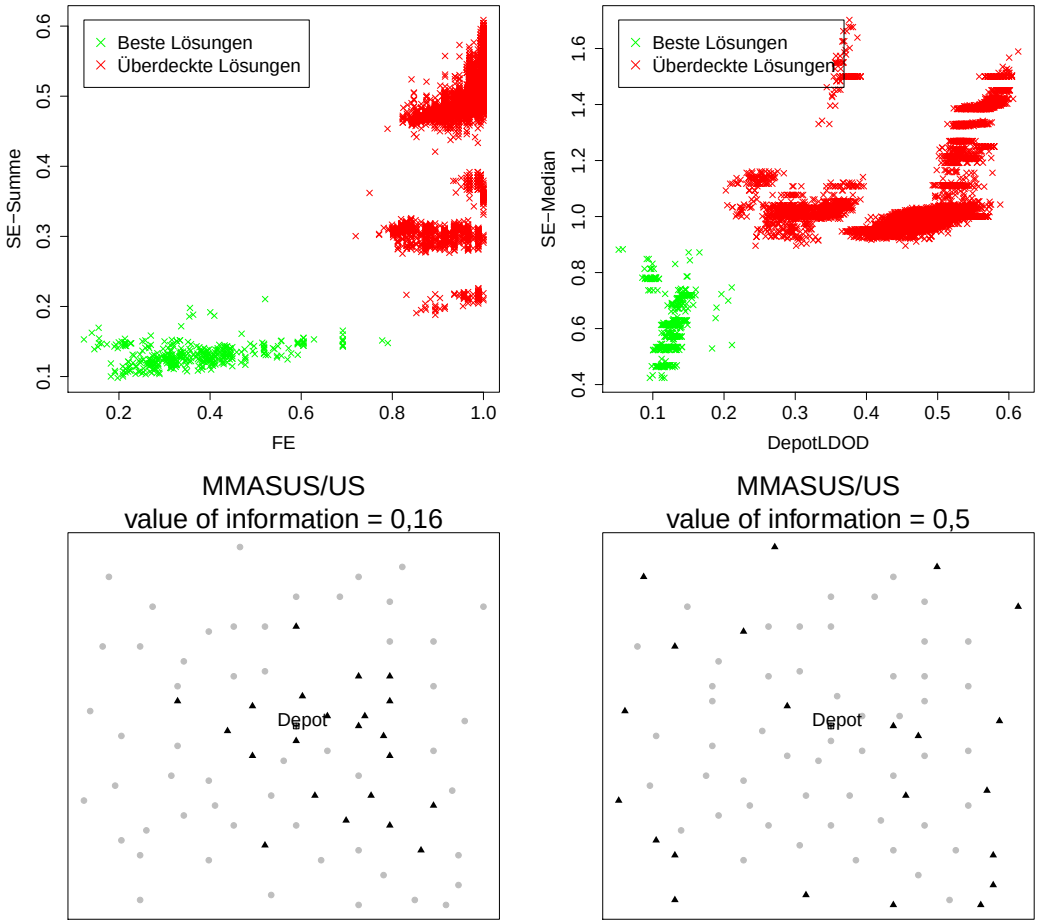

Abbildung 5.13: Beziehungen zwischen dem Algorithmus MMAS US/US und den vier einflussreichsten Problemeigenschaften, Visualisierung von Beispielinstanzen

Die in Abbildung 5.13 dargestellten Probleminstanzen sind aus den Mengen der untersuchten Instanzen zufällig aber mit gleicher zugrundeliegender statischer Probleminstanz (vgl. Abbildung 5.2) und einem DOD von 0,3 ausgewählt. Für die linke Instanz generiert kein anderer Algorithmus eine bessere Lösung als MMAS US/US. Für die rechte Instanz hingegen erzeugt der Algorithmus ausschließlich eine überdeckte Lösung. Die abgebildeten Probleminstanzen bestätigen visuell die Erkenntnisse aus der Analyse der Graphen aus Abbildung 5.13. Befinden sich die dynamischen Kundenanfragen geballt um den geometrischen Schwerpunkt bzw. um das Depot zeigt der Algorithmus seine Stärken und generiert beste Lösungen. Verteilen sich die dynamischen Anfragen bis an den Rand der Probleminstanz werden generierte Lösungen durch andere Algorithmen überdeckt. Bei dem Vergleich der abgebildeten Probleminstanzen mit den

generierten Instanzen aus Unterabschnitt 4.1.3 kann der Zusammenhang zwischen den Ausprägungen der Problemeigenschaften und der Beschaffenheit der Probleminstanz nachvollzogen werden.

NN 2-Opt/Insert

Der NN 2-Opt/Insert-Algorithmus verhält sich ähnlich wie MMAS US/US, zeigt sich aber robuster hinsichtlich der Ausprägungen der Eigenschaften. Beste Lösungen werden auch noch für höhere Werte der Eigenschaften *FE* und SE_{Median} durch den Algorithmus generiert ($> 0{,}8$). Vereinzelt werden auch noch beste Lösungen für einen mittleren *DepotLDOD* und SE_{Summe} gefunden (vgl. Abbildung 5.14). An den Graphen in Abbildung 5.14 ist auch zu erkennen, dass die Mengen der besten und überdeckten Lösungen enger zusammenrücken und es teilweise zu Überschneidungen kommt. So können für große Werte der Eigenschaft *FE* und kleine Werte für SE_{Summe} sowohl beste als auch überdeckte Lösungen gefunden werden. Ähnliches gilt für mittlere Werte für SE_{Median} und geringe bis mittlere Werte für *DepotLDOD*. Werden mehr als 20 Prozent der jeweils besten bzw. überdeckten Lösungen visualisiert, überschneiden sich die Mengen noch stärker. Eine klare Trennung ist dann nur mithilfe der vier einflussreichsten Eigenschaften nicht mehr möglich. Die Mengen der überdeckten Lösungen für die Algorithmen MMAS US/US und NN 2-Opt/Insert sind sich ähnlich. Das bestätigt das Ergebnis der Klassifikation der Algorithmen nach der Distanz der generierten Lösungen, durchgeführt in Unterabschnitt 5.1.3. Der Abstand der Klassen, in denen sich die Algorithmen NN 2-Opt/Insert und MMAS US/US befinden, ist am niedrigsten im Vergleich zu den Abständen zwischen den anderen Klassen (vgl. Abbildung 5.11).

Die auf die gleiche Weise wie für den MMAS US/US-Algorithmus ausgewählten Probleminstanzen, abgebildet in der zweiten Zeile der Abbildung 5.14, bestätigen die Ähnlichkeit der Algorithmen. Die Beispielinstanzen für die besten bzw. die überdeckten Lösungen weisen identische Charakteristika auf. Ballen sich die dynamischen Kundenanfragen um den geometrischen Schwerpunkt, generiert NN 2-Opt/Insert beste Lösungen, verteilen sich die Anfragen hingegen weiter am Rand, kann der Algorithmus nur noch überdeckte Lösungen ermitteln. Eine Auswahl zwischen den Algorithmen NN 2-Opt/Insert und MMAS US/US basierend auf der reinen Visualisierung der Probleminstanzen scheint nicht möglich zu sein. Es zeigt sich aber in dieser ersten Analyse, dass dynamische Kundenanfragen am Rand der Probleminstanz ungünstig sowohl für den MMAS US/US- als auch für den NN 2-Opt/Insert-Algorithmus sind (mittlere bis große Werte für die einflussreichsten Eigenschaften). Sind die Werte für die vier wichtigsten Eigenschaften besonders klein, ist der MMAS US/US- gegenüber dem NN 2-Opt/Insert-Algorithmus zu bevorzugen. Die Ergebnisse dieser Analyse werden im Folgenden mit der Visualisierung der Wertebereiche mithilfe von Box-Plots bestätigt (vgl. Abbildung 5.19).

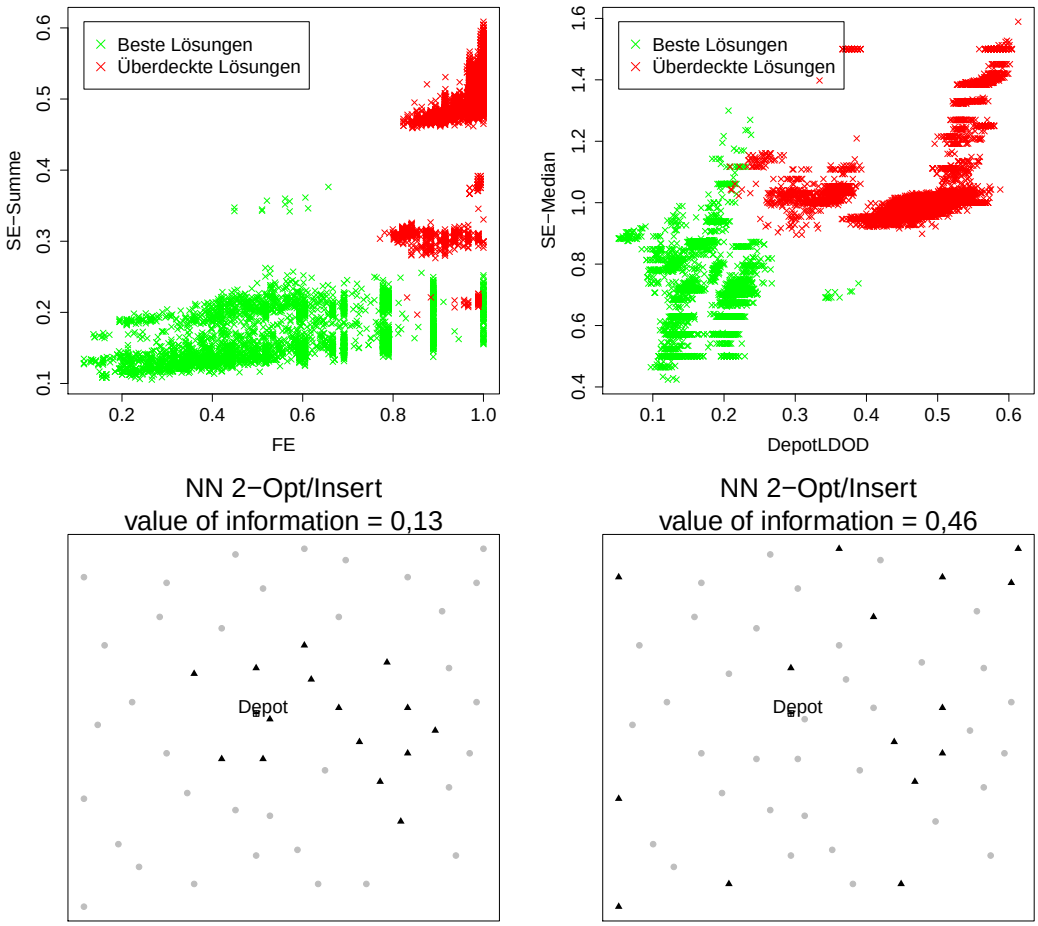

Abbildung 5.14: Beziehungen zwischen dem Algorithmus NN 2-Opt/Insert und den vier einflussreichsten Problemeigenschaften, Visualisierung von Beispielprobleminstanzen

Spiral/Insert

Die Hauptmotivation für die Erarbeitung des Spiralalgorithmus ist die Erkenntnis, dass die betrachteten Algorithmen dazu tendieren statische Kunden im äußeren Bereich zuerst zu bedienen und Kunden in einem Bereich als Nachbarn in eine Rundreise einzuplanen (vgl. Unterabschnitt 4.2.7). Aus diesem Grund können dynamische Kunden an den Randbereichen nur unter erheblichen Mehrkosten eingeplant werden (vgl. Abbildung 4.47 in Unterabschnitt 4.2.7). Die eben diskutierten Ergebnisse, visualisiert in Abbildung 5.13 und Abbildung 5.14, bestätigen die in Unterabschnitt 4.2.7 formulierten Erkenntnisse. Die Darstellung der Beziehungen zwischen den Problemeigenschaften und den Mengen der besten und überdeckten Lösungen für den Spiral/Insert-Algorithmus, visualisiert in Abbildung 5.15, zeigt, dass der Spiral/Insert-Algorithmus bei hohen Werten für FE, SE_{Summe}, SE_{Median} und $DepotLDOD$ beste Lösungen generiert. Damit bestätigt sich die These 4.2.1 aufgestellt in Unterabschnitt 4.2.7, wonach die Lösungsqualität einer flexibel geplanten Rundreise für einen mittleren DOD und einen hohen Wert für die Flächen-

Eigenschaft *FE* hoch ist. Zusätzlich zeigen die Graphen, dass die Mengen der besten und überdeckten Lösungen sich stark überschneiden (vgl. Abbildung 5.15). Eine einfache lineare Trennung zwischen den Mengen scheint auf der Basis der vier einflussreichsten Eigenschaften nicht möglich zu sein. Sowohl beste als auch überdeckte Lösungen werden für fast den gesamten Wertebereich der Eigenschaften *FE*, SE_{Summe} und *DepotLDOD* generiert. Die Abbildung 5.15 suggeriert, dass ausschließlich für geringe Werte für SE_{Median} ($< 0{,}7$) keine überdeckte Lösungen generiert werden. Unter Berücksichtigung der Abbildungen 5.13 und 5.14 kann dieser Darstellungseffekt aber auf die eingeschränkte Menge der überdeckten Lösungen zurückgeführt werden. Zusammenfassend zeigt sich in den Graphen der Abbildung 5.15, dass der Spiral/Insert-Algorithmus robuster im Vergleich zu den Algorithmen 2-Opt/Insert und MMAS US/US ist und auch für hohe Werte der Eigenschaften noch beste Lösungen generiert, aber auch über den gesamten Wertebereich der Eigenschaften nur überdeckte Lösungen generieren kann.

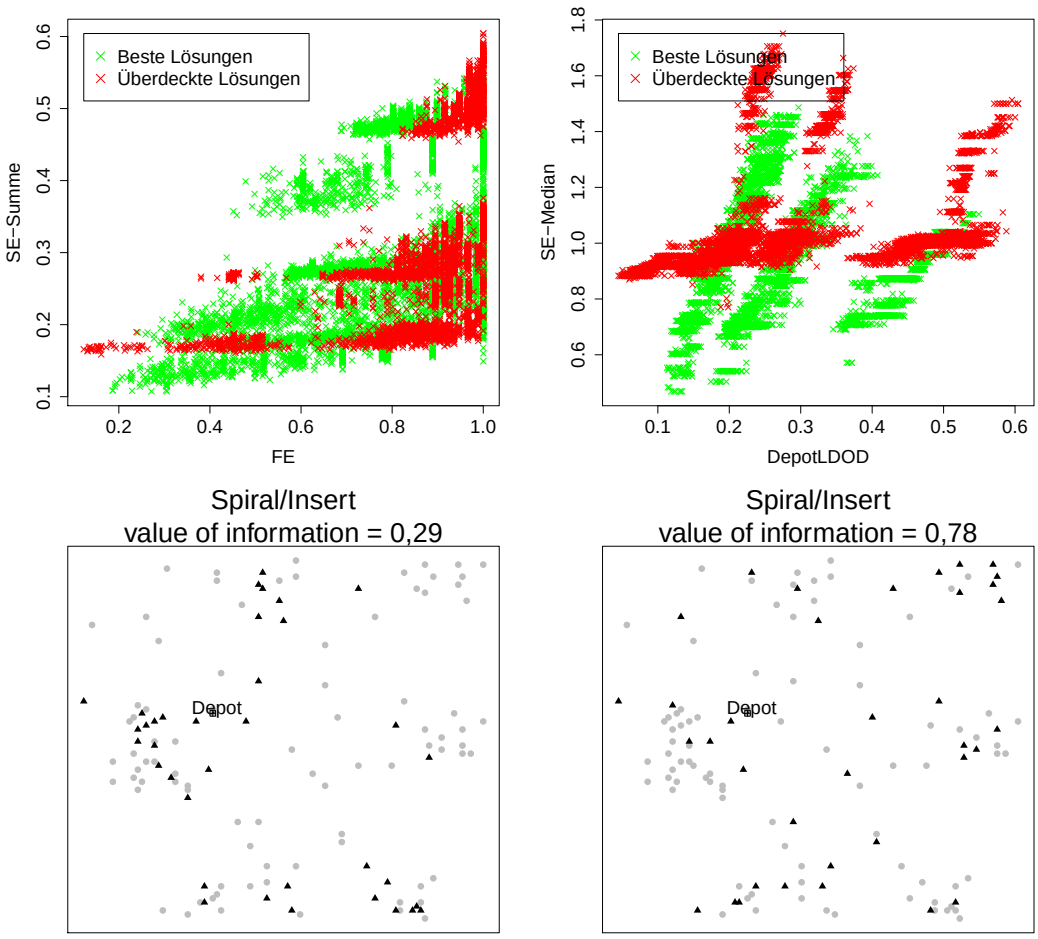

Abbildung 5.15: Beziehungen zwischen dem Algorithmus Spiral/Insert und den vier einflussreichsten Problemeigenschaften, Visualisierung von Beispielinstanzen.

Die Beispielinstanzen, abgebildet in der zweiten Zeile der Abbildung 5.15, bestätigen die diskutierten Erkenntnisse. Eine beste Lösung wird generiert, wenn die dynamischen

Anfragen weitläufiger und auch am Rand der Instanz verteilt sind (vgl. linke Instanz in Abbildung 5.15). In vielen generierten überdeckten Lösungen befinden sich alle dynamischen Kundenanfragen geballt im Zentrum der Probleminstanz (vgl. beste Probleminstanzen in Abbildung 5.13 oder Abbildung 5.14). Es existieren aber auch überdeckte Lösungen, bei denen die dynamischen Kundenanfragen noch weiter am Rand und verteilter auf der Instanz zu finden sind (vgl. rechte Instanz in Abbildung 5.15). Hier zeigt sich, dass der Spiralalgorithmus mit einer statischen Rastergröße im Vergleich zu einer dynamischen Rastergröße noch flexiblere Rundreisen plant. Eine Analyse zeigt, dass maximal 10% der initial bekannten Kundenanfragen in einer Rasterzelle zwar die besten Ergebnisse für zufällig generierte Probleminstanzen erzeugt (vgl. der Analyse in Unterabschnitt 4.2.7), aber für Instanzen mit, zum Beispiel, besonders hohen Werten für die Flächen-Eigenschaft *FE* teilweise nur überdeckte Lösungen generiert. Grund ist hierfür die zu kleine Rastergröße, die zu weniger Flexibilität bei der Lösung, durch den Planungsalgorithmus führt. Das bestätigt im Wesentlichen die Robustheit des Algorithmus hinsichtlich bester Lösungen für nahezu alle Werte im Wertebereich der einflussreichsten Problemeigenschaften.

Spiral8x8/Insert

Der Spiral8x8-Planungsalgorithmus verzichtet auf eine dynamische Bestimmung der Rastergröße. Das Raster besteht hier unabhängig von der zugrundeliegenden Instanz aus 64 Rasterzellen. Damit wird eine sehr lange aber auch sehr flexible Rundreise durch den Planungsalgorithmus geplant (auch im Vergleich zum Spiralplanungsalgorithmus). Die Visualisierung der Beziehungen der Eigenschaften und der Mengen der besten und überdeckten Lösungen zeigt das erwartete Muster. Beste Lösungen werden nun fast ausschließlich für große Werte von *FE* generiert ($> 0{,}4$). Zusätzlich werden keine besten Lösungen mehr für kleine Werte von SE_{Median} und *DepotLDOD* gefunden. Im Vergleich zu den vorherigen diskutierten Algorithmen erzeugt der Spiral8x8/Insert-Algorithmus beste Lösungen für die höchsten Ausprägungen der Problemeigenschaften. Bei kleineren Werten können nur noch überdeckte Lösungen generiert werden. Trotz der Planung der flexibelsten Rundreise kann der Algorithmus teilweise auch für hohe Werte der Eigenschaften nur überdeckte Lösungen generieren. Wie auch beim Spiral/Insert-Algorithmus zeigt sich, dass die Mengen, bestehend aus besten und überdeckten Lösungen, sich stark überlagern und nicht mehr linear voneinander getrennt werden können.

Die ausgewählten Beispielinstanzen für eine beste und eine überdeckte Lösung, gezeigt in der zweiten Reihe der Abbildung 5.16, bestätigen die diskutierten Erkenntnisse. Eine beste Lösung wird für Probleminstanzen gefunden, bei denen die dynamischen Anfragen weitläufig verteilt und an den Rändern der Instanz zu finden sind. Sind die Anfragen eher im Zentrum, wird nur eine überdeckte Lösung durch den Algorithmus generiert.

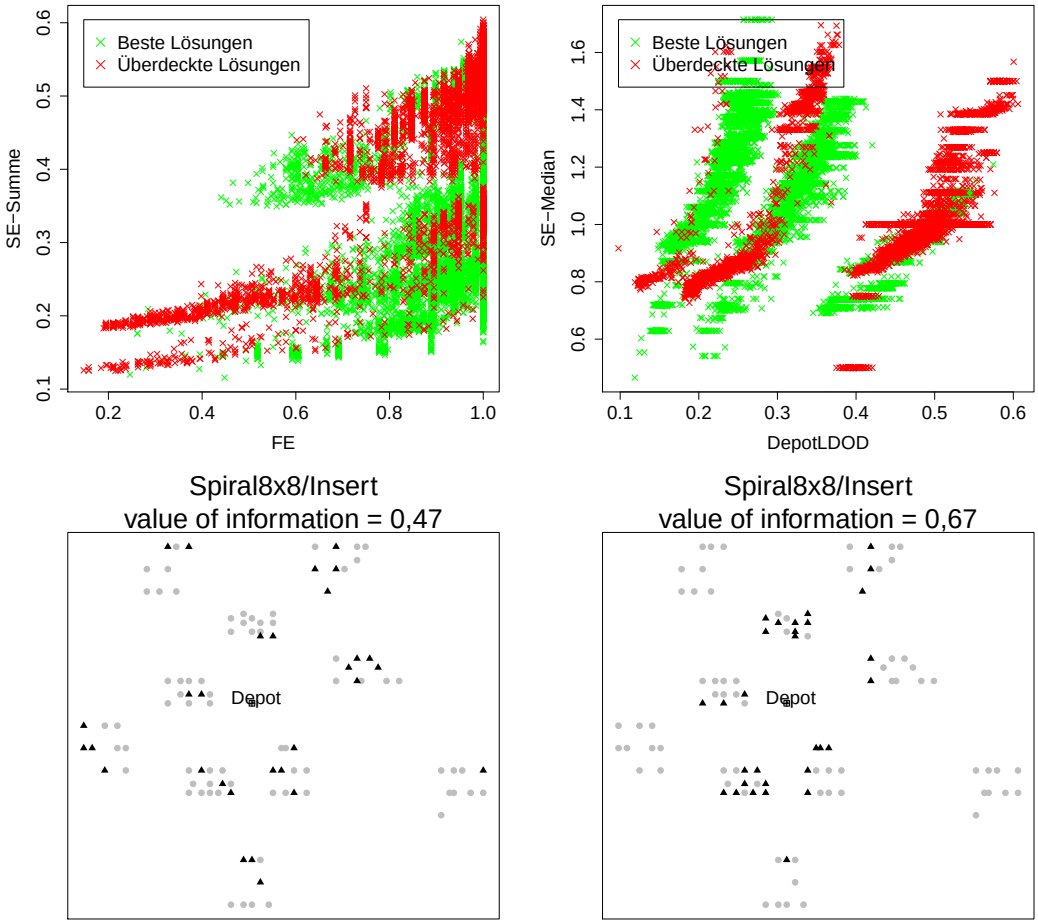

Abbildung 5.16: Beziehungen zwischen dem Algorithmus Spiral8x8/Insert und den vier einflussreichsten Problemeigenschaften, Visualisierung von Beispielinstanzen

Die nachfolgende Abbildung 5.17 bestätigt zusätzlich die eben besprochenen Ergebnisse und die These 4.2.1. In der linken Spalte der Abbildung sind die Ergebnisse der Algorithmen in Form des *Value of information* (vgl. Unterabschnitt 2.1.6) in Beziehung zu den Eigenschaften *FE* und *LDOD* für Probleminstanzen mit einem DOD von 0,3 dargestellt. In der rechten Spalte ist die Beziehung für alle Probleminstanzen abgebildet. Gut ist zu sehen, dass für große Werte von *FE* der Spiral8x8/Insert-Algorithmus die besten Lösungen erzeugt. Für kleine Werte hingegen ist der Algorithmus nicht konkurrenzfähig und wird von den Lösungen der anderen Algorithmen überdeckt. Der Spiral8x8/Insert-Algorithmus zeigt sich generell robust gegenüber Schwankungen der *FE*. Ein ähnlicher Effekt ist für den *LDOD* zu beobachten. Hier zeigt sich aber auch die besprochene Korrelation zwischen DOD und *LDOD*. Für Probleminstanzen mit einem DOD von 0,3 (untere Zeile, linkes Bild) ist der Zusammenhang zwischen hohem *LDOD* und bester Lösungen gut nachzuvollziehen. Bei der Betrachtungen aller Probleminstanzen sind starke Schwankungen des *Value of information* in Abhängigkeit des *LDOD* zu beobachten. Beste Lösungen werden hier ausschließlich für einen hohen

LDOD im Vergleich zum betrachteten DOD erzeugt. Mit den in Abbildung 5.17 gezeigten Ergebnissen bestätigt sich das Erreichen einer Zielstellung dieser Arbeit. Der Spiral8x8/Insert-Algorithmus ist sehr erfolgreich für Probleminstanzen mit bestimmten Ausprägungen von Eigenschaften.

Bei den Graphen in Abbildung 5.17 gilt generell zu beachten, dass hier ausschließlich ein gleitender Mittelwert über 5.000 Probleminstanzen sortiert nach der jeweiligen Eigenschaft abgebildet wird, um den Trend der Beziehungen von Eigenschaft und *Value of information* zu verdeutlichen. Bei der Abbildung aller Werte schwanken die Werte für den *Value of information* erheblich. Jeder betrachtete Algorithmus generiert beste Lösungen (vgl. Unterabschnitt 5.1.3).

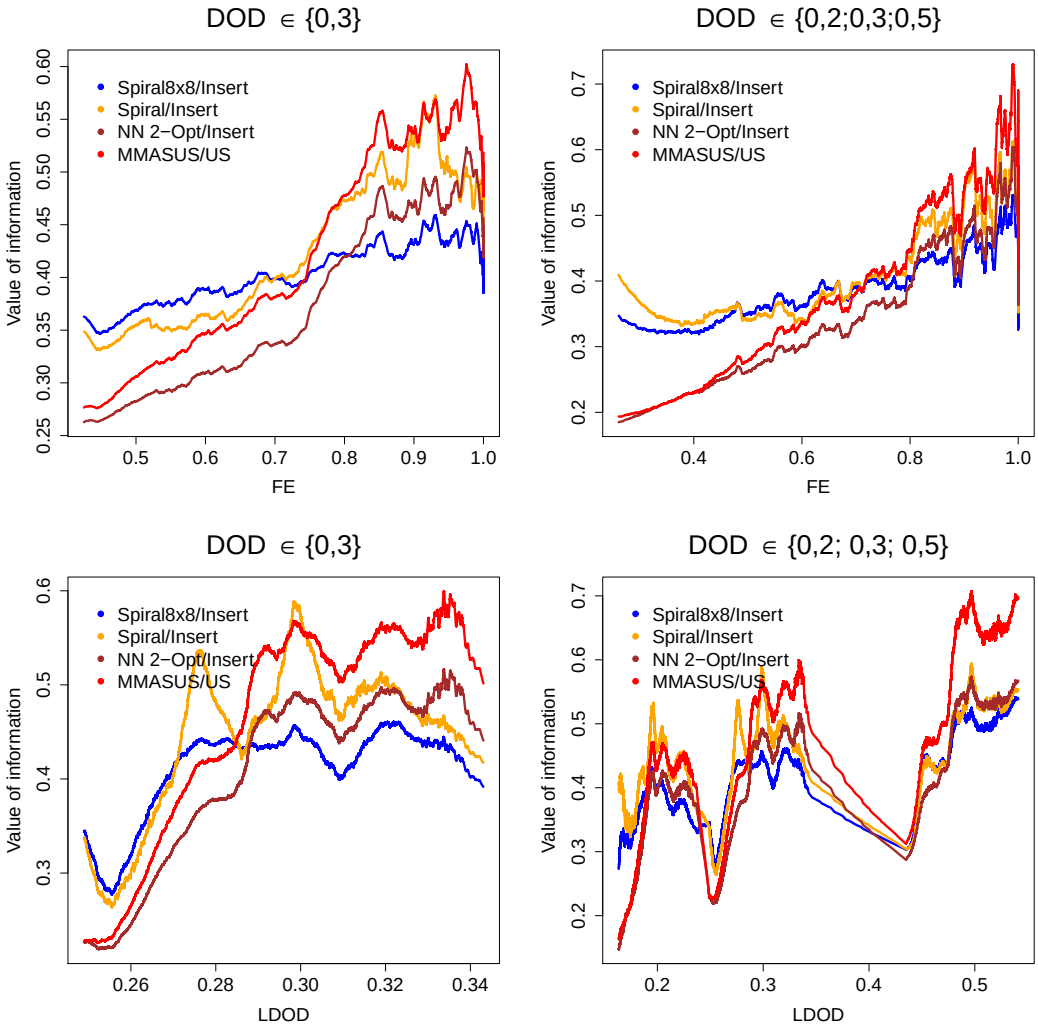

Abbildung 5.17: Beziehungen zwischen der Lösungsqualität und den Eigenschaften *FE* und *LDOD* für Instanzen mit einem DOD von 0,3 (linke Spalte) und für alle Probleminstanzen (rechte Spalte)

Zusammenfassung

Zusammenfassend sind die jeweils besten Lösungen der Algorithmen in Form der Beziehung zwischen den Eigenschaften SE_{Summe} - FE und SE_{median} - $DepotLDOD$ in Abbildung 5.18 dargestellt. Die Abbildung fasst die in den vorherigen Kapiteln besprochenen Erkenntnisse zusammen. Für größere werdende Werte der Eigenschaften erzeugen die Algorithmen in der Reihenfolge MMAS US/Insert, NN 2-Opt/Insert, Spiral/Insert und Spiral8x8/Insert die besten Lösungen.

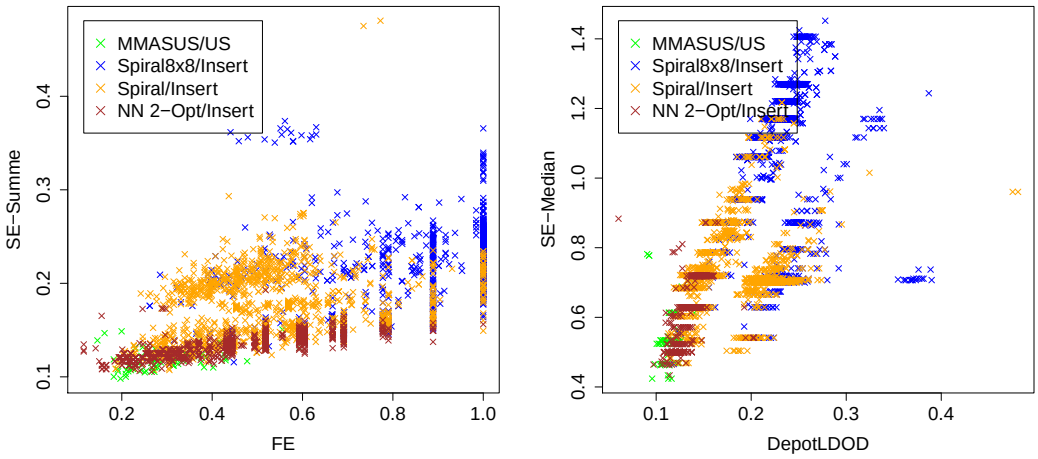

Abbildung 5.18: Beste Lösungen der Algorithmen in Form der Beziehung zwischen den Eigenschaften SE_{Summe} - FE und SE_{Median} - DepotLDOD

An den Box-Plots aus Abbildung 5.19 können die konkreten Wertebereiche der Eigenschaften der Instanzen, für die die Algorithmen beste Lösungen generieren abgelesen werden. Die Box-Plots stellen somit die Grundlage für eine manuelle Selektion der betrachteten Algorithmen basierend auf den vier einflussreichsten Eigenschaften dar. Eine manuelle Selektion von Algorithmen innerhalb einer identifizierten Klasse (vgl. Abbildung 5.11) ist aufgrund der Ähnlichkeit der Algorithmen anhand der vier einflussreichsten Eigenschaften nicht zielführend. Zu bedenken gilt zudem, dass die Wertebereiche nur 20% der besten Lösungen repräsentieren. Werden alle besten Lösungen in die Auswertung einbezogen, überdecken sich die Wertebereiche stark, sodass eine sinnvolle Aufteilung nicht mehr möglich ist. Zusammenfassend muss also festgestellt werden, dass eine manuelle Auswertung der Beziehung zwischen den Eigenschaften und der Problemschwere nur sehr rudimentäre Ergebnisse liefert. Einfache Beziehungen können aber beschrieben und näher charakterisiert werden. Die komplexen Zusammenhänge zwischen Problemschwere und Eigenschaften lassen sich aber nur schwer durch die Betrachtung der vier einflussreichsten Eigenschaften ergründen und visualisieren. Die Analyse der Beziehungen zwischen den Algorithmen und Eigenschaften beschränkt sich dennoch nur auf die vier wesentlichen Problemeigenschaften, da eine manuelle Selektion von Algorithmen auf der Grundlage von mehr Eigenschaften nicht praktikabel scheint.

Wie zuverlässig die automatisierte Selektion von Algorithmen basierend auf den vier einflussreichsten Eigenschaften ist, bzw. über wie viele wichtige Problemeigenschaften Prognosemodelle erstellt werden müssen, um erfolgreich automatisiert Algorithmen für Probleminstanzen auszuwählen, wird unter anderem im folgenden Kapitel diskutiert.

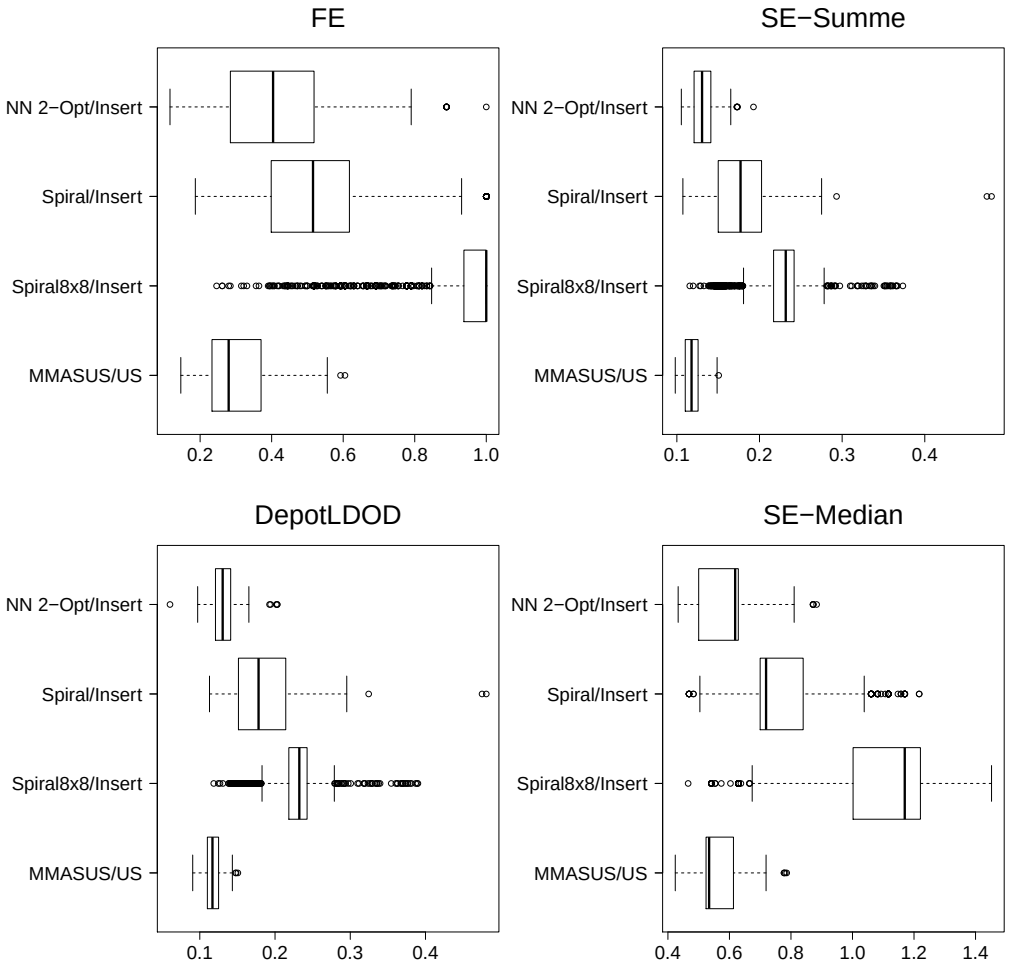

Abbildung 5.19: Wertebereiche der Eigenschaften für Probleminstanzen, für die die Algorithmen beste Lösungen generieren

5.3 Zusammenfassung

Mithilfe eines umfangreichen Data-Farming-Experiments werden unter Verwendung des in Abschnitt 5.1 eingeführten Probleminstanzgenerators schwer und leicht zu lösende Instanzen für die in dieser Arbeit betrachteten Algorithmen erzeugt. Dabei werden erstmalig wesentliche Eigenschaften von Algorithmen für DVRP erarbeitet, die bei einer Generierung von repräsentativen Lösungen für Problemstellungen berücksichtigt werden müssen. Anhand der generierten und gelösten Instanzen zeigt sich, dass die betrachteten Algorithmen komplementär zueinander sind. Es werden eine Vielzahl

von Instanzen identifiziert, für die verschiedene Algorithmen beste Lösungen erzeugen. Damit wird nachgewiesen, dass eine Selektion von Algorithmen notwendig und zielführend ist. Zudem kann so gezeigt werden, dass ein fairer Vergleich von Algorithmen nur auf der Grundlage der in dieser Arbeit eingeführten Problemeigenschaften möglich ist (vgl. Unterabschnitt 5.1.3). Es existieren Probleminstanzen für die Algorithmen ganz unterschiedliche Qualitäten von Lösungen erzeugen. Nur der NN 2-Opt/US-Algorithmus findet ausschließlich überdeckte Lösungen und wird aus diesem Grund bei den weiterführenden Analysen nicht mehr berücksichtigt. Mithilfe einer Klassifikation der Algorithmen nach dem Abstand der erzeugten Lösungen, werden Gruppen von ähnlichen Algorithmen identifiziert. Für je einen Vertreter einer Gruppe werden weiterführenden Analysen der Beziehungen zwischen Eigenschaften und Lösungen vorgenommen. Dabei wird erstmalig der Nachweis geführt, dass sich die in dieser Arbeit eingeführten Problemeigenschaften (vgl. Kapitel 4) für die Unterscheidbarkeit von Probleminstanzen eignen. Zusätzlich werden die für die Trennung zwischen leichten und schweren Problemstellungen wichtigsten Eigenschaften identifiziert. Für die einflussreichsten vier identifizierten Problemeigenschaften werden die Beziehungen zwischen schweren bzw. leichten Instanzen für die Algorithmen untersucht. Die Ergebnisse der Untersuchung sind Wertebereiche der Eigenschaften, anhand derer eine manuelle Auswahl zwischen den Vertretern der Gruppen der Algorithmen möglich ist. Es wird in diesem Kapitel also die Voraussetzung dafür geschaffen, dass gewonnene Erkenntnisse, auch unabhängig von im Zuge dieser Arbeit implementierten Modellen, für die automatisierte Auswahl von Algorithmen, genutzt werden können.

Die durch das Data-Farming-Experiment gewonnenen und für die Algorithmen repräsentativen Probleminstanzen bilden die Grundlage für die Implementierung der Modelle (Optimierer, vgl. Abbildung 5.1) für die automatisierte Auswahl von Algorithmen. Aus den abgeleiteten Erkenntnissen ergeben sich zusätzlich weitere Problemstellungen. Zum Beispiel, wird im folgenden Kapitel erörtert, wie erfolgreich die automatisierte Auswahl von Algorithmen auf der Basis von nur vier der wichtigsten Problemeigenschaften ist, und wie viele Eigenschaften für eine präzise Auswahl benötigt werden. Zusätzlich wird in einer umfangreichen Diskussion in Abschnitt 6.2 erläutert, inwieweit das durchgeführte Experiment und die Auswahl der Parameter Einfluss auf den Erfolg der automatisierten Selektion von Algorithmen haben.

Kapitel 6

Selektion von Algorithmen

Das folgende Kapitel diskutiert die Erstellung und Anwendung von Modellen $S(f(x))$ (vgl. Unterabschnitt 2.2.4) für die automatisierte Auswahl von Algorithmen. Für die Einordnung des Kapitels in den Kontext der Arbeit ist in Abbildung 6.1 der Optimierer als Generalisierung der Modelle für die automatisierte Auswahl von Algorithmen grau hervorgehoben.

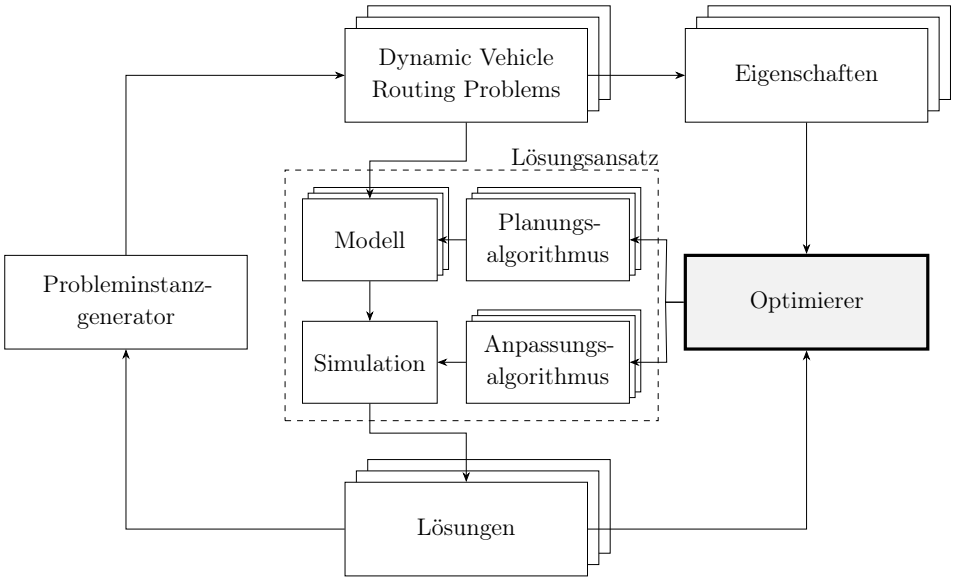

Abbildung 6.1: Lösungsansatz mit Fokus auf der Komponente Optimierer

In Abbildung 6.1 ist zu sehen, dass ein Optimierer sowohl Lösungen, als auch Eigenschaften verarbeitet, um Algorithmen auszuwählen. In einem ersten Schritt wird der Zusammenhang zwischen den Eigenschaften und den zugehörigen Lösungen modelliert. In einem zweiten Schritt können die Modelle für die automatisierte Auswahl von Algorithmen angewendet werden. Voraussetzung für die Umsetzung der beiden Schritte ist die Exploration und Analyse von Problemstellungen und Lösungen, erarbeitet in Kapitel 5. Hier wird die Datenbasis als Grundlage für die Umsetzung des ersten Schrittes gelegt. In der Literatur werden verschiedene Ansätze für die Implementierung von Selektionsmodellen $S(f(x))$ diskutiert (vgl. Unterabschnitt 2.2.4). In Unterabschnitt 6.1.1

werden zwei der diskutierten Ansätze umgesetzt und die Ergebnisse des ersten Schrittes vorgestellt. Unterabschnitt 6.1.2 demonstriert dann die erfolgreiche Selektion von Algorithmen mithilfe der eingeführten Selektionsmodelle. Damit wird erstmalig der Nachweis geführt, dass die Selektion von Algorithmen auf Basis der in dieser Arbeit eingeführten Problemeigenschaften automatisiert realisierbar ist. In Unterabschnitt 6.1.3 wird das erfolgreichere der beiden eingeführten Modelle noch zusätzlich mithilfe von Problemstellungen evaluiert, die unabhängig von den in Abschnitt 5.1 vorgestellten statischen Probleminstanzen sind. Solche dynamischen Problemstellungen werden im Folgenden als unbekannte Probleminstanzen bezeichnet. Mit einer Zusammenfassung und einer ausführlichen Diskussion der Ergebnisse schließt das Kapitel. Da der Erfolg der Selektionsmodelle stark von der geschaffenen Datenbasis abhängig ist, werden in der Diskussion auch Aspekte aus Kapitel 5 adressiert.

6.1 Automatisierte Selektion von Algorithmen

Unterabschnitt 2.2.4 diskutiert verschiedene Ansätze für die Selektion von Algorithmen. Im folgenden Unterabschnitt 6.1.1 werden die zwei in dieser Arbeit umgesetzten Modelle $S(f(x))$ vorgestellt und miteinander verglichen. Das erste erstellte Modell $S_{reg}(f(x))$ beruht auf einem von Xu et al. (2008) vorgestellten und für viele Problemstellungen etablierten Modell $S(f(x))$ und wird aus diesem Grund in dieser Arbeit betrachtet. Das zweite betrachtete Modell $S_{p-reg}(f(x))$ erweitert eine Weiterentwicklungen des ersten Modells von Xu et al. (2008). Auch diese Weiterentwicklung, eingeführt von Xu et al. (2011), hat sich für verschiedene Problemstellungen etabliert und wird aus diesem Grund in dieser Arbeit betrachtet (vgl. Unterabschnitt 2.2.4). Unterabschnitt 6.1.1 erörtert ebenfalls die Trainingsergebnisse der umgesetzten Regressionsmodelle, auf deren Grundlage ein Selektionsmodell $S(f(x))$ die Entscheidung für einen Algorithmus trifft. In Unterabschnitt 6.1.2 wird der Nachweis geführt, dass eine erfolgreiche Selektion von Algorithmen auf der Basis der in dieser Arbeit eingeführten Eigenschaften mithilfe der erarbeiteten Modelle möglich ist. Eine Evaluation der erfolgreichen Selektionsmodelle mithilfe von unbekannten Probleminstanzen wird in Unterabschnitt 6.1.3 durchgeführt.

Alle Regressionsmodelle werden in dieser Arbeit mithilfe der quelloffenen Bibliothek Keras für neuronale Netze umgesetzt. Keras wurde von Chollet et al. (2015) veröffentlicht und stellt die Schnittstellen zu der quelloffenen Machine-Learning-Bibliothek TensorFlow, veröffentlicht von Abadi et al. (2016) (Google-Brain), zur Verfügung. In den durchgeführten Experimenten nutzt TensorFlow für die Berechnung der Modelle die Grafikkarte GeForce GTX 1070 von NVIDIA. Das Training eines implementierten Regressionsmodells nimmt unter den gegebenen Rahmenbedingungen rund eine Stunde in Anspruch. Für die Realisierung der beiden vorgestellten Ansätze werden insgesamt 66 Regressionsmodelle implementiert.

6.1.1 Modelle für die Selektion von Algorithmen

Eine der ersten erfolgreichen Implementierungen für ein Modell $S(f(x))$ für das SAT, erläutert in Unterabschnitt 2.2.1, basiert nach Wagner et al. (2018) auf empirischen Modellen, die die Performance der Algorithmen vorhersagen (vgl. Unterabschnitt 2.2.4). Auf der Grundlage der Vorhersagen über die zu erwartende Performance wird eine Entscheidung für einen Algorithmus getroffen. Einen ähnlichen Ansatz verfolgt das in dieser Arbeit umgesetzte Modell $S_{reg}(f(x))$ formuliert in Gleichung 6.1.

$$S_{reg}\big(f(x)\big) = argmin\Big(N_{\alpha_1}\big(f(x)\big), \ldots, N_{\alpha_n}\big(f(x)\big)\Big) \tag{6.1}$$

Für jeden Algorithmus α_i wird ein Regressionsmodell $N_{\alpha_i}(f(x))$ erstellt, das die Beziehungen zwischen den Eigenschaften und der Lösungsqualität des Algorithmus abbildet. Derjenige Algorithmus wird selektiert, für den das zugehörige Modell die höchste Qualität der Lösung voraussagt. Der selektierte Algorithmus wird für die Bearbeitung der gegebenen Problemstellung herangezogen. Die höchste Lösungsqualität entspricht dem niedrigsten vorhergesagten erweiterten Value of Information $V_A^*(I)$ (vgl. Unterabschnitt 2.1.6).

Für das Training der Regressionsmodelle werden die in Abschnitt 5.1 beschriebenen Daten herangezogen. Außer für die Algorithmen Spiral/Insert und NN 2-Opt/US werden die kompletten, für die jeweiligen Algorithmen erzeugten und gelösten Probleminstanzen für den Trainings- bzw. Testprozess verwendet. So stehen für die Generierung der Regressionsmodelle, für jeden der Algorithmen MMASUS/Insert, MMASUS/US, Spiral8x8/Insert, NN 2-Opt/Insert und NN 2-Opt/NN 2-Opt mindestens 60.000 gelöste Probleminstanzen zur Verfügung. Für die von allen Algorithmen gelösten Probleminstanzen, erzeugt für die Algorithmen Spiral/Insert und NN 2-Opt/US, werden nur 60% für den Trainings- bzw. Testprozess für die Regressionsmodelle für Spiral/Insert und NN 2-Opt/US verwendet. Die nicht für das Erstellen der Regressionsmodelle herangezogenen Daten werden für den Nachweis der erfolgreichen Selektion von Algorithmen eingesetzt, der in Unterabschnitt 6.1.2 im Detail vorgestellt wird. Eine Standardisierung der Eingabe- und Ausgabedaten wird nicht vorgenommen. Der Wertebereich des erweiterten Value of Information ist bereits unabhängig von einer konkreten Probleminstanz (vgl. Unterabschnitt 2.1.6). Im Rahmen dieser Arbeit durchgeführte Experimente zeigen, dass das Erzeugen der Modelle mithilfe von standardisierten Eingabe- bzw. Ausgabedaten zu schlechteren Ergebnissen bei der Selektion von Algorithmen führt. Für jeden Algorithmus werden drei verschiedene Regressionsmodelle erzeugt. Ein Modell berücksichtigt ausschließlich die erarbeiteten vier wichtigsten Problemeigenschaften, ein weiteres Modell berücksichtigt die acht wichtigsten und eines sämtliche Problemeigenschaften (vgl. Abbildung 5.12). So soll experimentell untersucht werden, ob auch mit dem Einsatz von wenigen Eigenschaften ein erfolgreiches Selektionsmodell umzusetzen ist. Die Regressionsmodelle werden mithilfe von künstlichen neuronalen Netzen realisiert (vgl. Unterabschnitt 2.2.4). Die Struktur der neuronalen Netze wird mithilfe eines systematischen Versuch-und-Irrtum-Ansatzes erarbeitet. Dieser Ansatz ist nach wie vor

üblich bei der Erstellung von künstlichen neuronalen Netzen (Alvarez und Salzmann, 2016). Die erfolgreichste Struktur für Netze, die die vier wichtigsten Problemeigenschaften verarbeiten, besteht aus fünf verborgenen Schichten mit 40, 16, 10, 11 und 12 voll verbundenen Neuronen. Die Eingabeschicht hat eine Dimension von vier, aufgrund der Zahl der betrachteten Eigenschaften. Bis auf die Ausgabeschicht, die die Eingabewerte direkt ausgibt, verwenden alle Neuronen in den Schichten die Aktivierungsfunktion RELU, eingeführt von Nair und Hinton (2010). Das Optimierungsverfahren „Adam“, eingeführt von Kingma und Ba (2014), wird für die Bestimmung der Gewichte der Kanten im neuronalen Netz eingesetzt. Die Struktur der Regressionsmodelle, die die acht wichtigsten und alle Problemeigenschaften verarbeiten, besteht ebenfalls aus fünf verborgenen Schichten mit 60, 18, 20, 22 und 24 Neuronen. Die Eingabeschichten haben eine Dimension von acht, bzw. von 181 (definiert durch die Anzahl der betrachteten Problemeigenschaften). Als Aktivierungsfunktion kommt hier ebenfalls RELU und als Optimierungsalgorithmus „Adam“ zum Einsatz.

Alle implementierten künstlichen neuronalen Netze werden mithilfe einer Kreuzvalidierung evaluiert. Die Ergebnisse der Validierung sind in Tabelle 6.1 abgebildet.

Tabelle 6.1: Ergebnisse der Kreuzvalidierung (5 Gruppen) der Regressionsmodelle

Algorithmus	Mittlerer quadratische Fehler (+/- Standardabweichung)		
	4 wichtigsten Eigenschaften	8 wichtigsten Eigenschaften	Alle Eigenschaften
MMASUS/Insert	0,00715 (+/- 0,00169)	0,00071 (+/- 0,00009)	0,00064 (+/- 0,00008)
MMASUS/US	0,00948 (+/- 0,00253)	0,00153 (+/- 0,00057)	0,00137 (+/- 0,00054)
Spiral8x8/Insert	0,01015 (+/- 0,00481)	0,00178 (+/- 0,00010)	0,00134 (+/- 0,00009)
Spiral/Insert	0,01865 (+/- 0,00273)	0,00539 (+/- 0,00106)	0,00340 (+/- 0,00023)
NN 2-Opt/Insert	0,00782 (+/- 0,00169)	0,00093 (+/- 0,00013)	0,00079 (+/- 0,00008)
NN 2-Opt/NN 2-Opt	0,00813 (+/- 0,00044)	0,00060 (+/- 0,00002)	0,00085 (+/- 0,00042)
NN 2-Opt/US	0,01272 (+/- 0,00324)	0,00168 (+/- 0,00033)	0,00166 (+/- 0,00054)

Die Methode der Kreuzvalidierung wird im Folgenden kurz anhand der Anwendung in dieser Arbeit beschrieben. Weiterführende Informationen sind, zum Beispiel, in Witten et al. (2016) zu finden. Für die Kreuzvalidierung werden die verwendeten Daten in fünf gleich große Gruppen eingeteilt. Alle betrachteten DOD und statische Probleminstanzen (vgl. Abbildung 5.2) werden gleichmäßig auf die Gruppen verteilt. Die Daten von jeweils

vier der fünf Gruppen werden zum Training der Regressionsmodelle verwendet. Mit der fünften Gruppe wird das erzeugte Netz evaluiert. Als Evaluationsmetrik wird der mittlere quadratische Fehler verwendet. Jedes Regressionsmodell wird mit fünf verschiedenen Teilmengen, die sich aus verschiedenen Kombinationen der fünf Gruppen zusammensetzen, trainiert und evaluiert. Der Durchschnitt der gemessenen mittleren quadratischen Fehler und die resultierende Standardabweichung sind als Ergebnisse der Evaluation in Tabelle 6.1 erfasst. An den durchschnittlichen mittleren quadratischen Fehlern aus Tabelle 6.1 ist unter anderem gut zu erkennen, dass die Beziehungen zwischen Problemeigenschaften und Lösungsqualität für alle Algorithmen annähernd mit gleicher, hoher Qualität bestimmt werden können. Auch ist zu sehen, dass die Vorhersagegenauigkeit mit der Anzahl der verwendeten Problemeigenschaften korreliert. Ein Modell, das acht Problemeigenschaften verarbeitet, erreicht eine wesentliche Verbesserung der Vorhersagegenauigkeit im Vergleich zu einem Modell, dass nur vier Eigenschaften berücksichtigt. Modelle, die alle Eigenschaften verarbeiten sind nur minimal besser als solche, die acht Eigenschaften betrachten. Für eine präzise Vorhersage der Lösungsqualität eines Algorithmus reichen demzufolge die acht wichtigsten Problemeigenschaften aus.

Das zweite in dieser Arbeit betrachtete Modell $S_{p-reg}(f(x))$, abgebildet in Gleichung 6.2, nutzt paarweise Regressionsmodelle für die Selektion von Algorithmen. Für jede Zweierkombination der n Algorithmen wird ein Regressionsmodell $N_{\alpha_a,\alpha_b}(f(x))$ erarbeitet, das die Beziehungen der Differenz der zu erwartenden Lösungsqualität der beiden Algorithmen α_a und α_b und den Problemeigenschaften modelliert. Die Summen der Lösungsqualitäten $\sum_{i=1}^{n} N_{\alpha_x,\alpha_i}$ für die Algorithmen wird als Entscheidungskriterium für die Selektion von Algorithmen herangezogen. Jedes paarweise Modell wählt demzufolge zwischen zwei Algorithmen. Ist die Summe negativ, bedeutet das, der Algorithmus α_a erzeugt eine höhere Lösungsqualität im Vergleich zum Konkurrenzalgorithmus α_b. Ist die Summe positiv, erwartet das Modell eine höhere Lösungsqualität für den Konkurrenzalgorithmus α_b. Die erarbeiteten paarweise Regressionsmodelle sind eine Erweiterung der paarweisen Klassifikationsmodelle eingeführt in Xu et al. (2011) (vgl. Unterabschnitt 2.2.4). In der Arbeit von Xu et al. (2011) werden paarweise Klassifikationsmodelle für die Selektion von Algorithmen eingeführt. Das Ergebnis dieser Modelle beschränkt sich auf das Benennen des besseren Algorithmus. Mithilfe eines paarweisen Regressionsmodells kann zusätzlich die Differenz der Lösungsqualität bestimmt werden.

$$S_{p-reg}\big(f(x)\big) = argmin\Bigg(\sum_{i=1}^{n} N_{\alpha_1,\alpha_i}\big(f(x)\big), \ldots, \sum_{i=1}^{n} N_{\alpha_n,\alpha_i}\big(f(x)\big)\Bigg) \tag{6.2}$$

Insgesamt werden 15 paarweise Regressionsmodelle mithilfe von künstlichen neuronalen Netzen für jede Kombination von Algorithmen realisiert. Für den von allen anderen Algorithmen überlagerten NN 2-Opt/US-Algorithmus werden keine Modelle konstruiert (vgl. Unterabschnitt 5.1.3). Der NN 2-Opt/US-Algorithmus wird demzufolge bei der Selektion durch das Modell $S_{p-reg}(f(x))$ nicht berücksichtigt. Für das Training und die Evaluation der paarweisen Modelle werden 60% der Probleminstanzen herangezogen,

die für den NN 2-Opt/US- und Spiral/Insert-Algorithmus generiert und von allen Algorithmen gelöst wurden. Hierbei handelt es sich um dieselbe Menge von Probleminstanzen, die auch für das Training der Regressionsmodelle für den NN 2-Opt/US- bzw. Spiral/Insert-Algorithmus verwendet werden. Für das Training und die Evaluation für jedes paarweise Regressionsmodell stehen demzufolge mindestens 72.000 verschiedene Probleminstanzen zur Verfügung. Wie auch für die Regressionsmodelle für $S_{reg}(f(x))$ wird von einer Standardisierung der Eingabe- bzw. Ausgabedaten abgesehen. Die Struktur der paarweisen Regressionsmodelle unterscheidet sich nicht von der beschriebenen Struktur der Regressionsmodelle für $S_{reg}(f(x))$. Tabelle 6.2 zeigt die Ergebnisse der Kreuzvalidierung mit fünf Gruppen für die erzeugten paarweisen Regressionsmodelle. Bei der Auswahl der Daten für die Gruppen der Kreuzvalidierung wird darauf geachtet, dass alle statischen Probleminstanzen und DOD in allen Gruppen vertreten sind.

Die Ergebnisse aus Tabelle 6.2 zeigen ein ähnliches Bild wie die Ergebnisse der Regressionsmodelle für $S_{reg}(f(x))$ aus Tabelle 6.1. Die Differenzen zwischen den Algorithmen können sehr gut mithilfe der Problemeigenschaften abgebildet werden. Mit nur vier Problemeigenschaften sind die Abbildungen am ungenausten und nehmen leicht mit der Anzahl der Eigenschaften zu. Im Unterschied zu den Regressionmodellen für $S_{reg}(f(x))$ ist die Verbesserung, die durch das Verwenden von allen Eigenschaften im Gegensatz den besten acht, erreicht wird, signifikant. Die sehr geringen Standardabweichungen zeigen, dass die Modelle unabhängig von den zugrunde liegenden Trainingsdaten sind.

Beide implementierten Ansätze werden in der Literatur verfolgt und haben sich mehrfach bewährt, siehe, zum Beispiel, Wagner et al. (2018). Der Vorteil des Modells $S_{reg}(f(x))$ liegt in der Erweiterbarkeit. Nur ein zusätzliches Regressionsmodell muss implementiert werden, um einen weiteren Algorithmus in das Portfolio für die Selektion aufzunehmen. Soll ein weiterer Algorithmus für das Modell $S_{p-reg}(f(x))$ betrachtet werden, muss für jeden bereits vorhandenen Algorithmus ein zusätzliches paarweises Modell erstellt werden. Die Kriterien für die Algorithmen, die für die Entscheidungsfindung herangezogen werden, beruhen hier auf mehreren Modellen, was das Vertrauen in ein Kriterium für einen Algorithmus und somit für die Entscheidung erhöht. Ein Kriterium für einen Algorithmus im Modell $S_{reg}(f(x))$ wird ausschließlich von einem Regressionsmodell bestimmt. Ein Modell mit nur einer geringen Vorhersagegenauigkeit kann sich hier wesentlicher auf das Ergebnis der Selektion auswirken.

Tabelle 6.2: Ergebnisse der Kreuzvalidierung (5 Gruppen) der paarweisen Regressionsmodelle für die Probleminstanzen (für NN 2-Opt/US erzeugt und durch alle Algorithmen gelöst)

Algorithmus	Mittlerer quadratische Fehler (+/- Standardabweichung)		
	4 wichtigsten Eigenschaften	8 wichtigsten Eigenschaften	Alle Eigenschaften
MMASUS/Insert - MMASUS/US	0,00187 (+/- 0,00048)	0,00109 (+/- 0,00004)	0,00047 (+/- 0,00005)
MMASUS/Insert - Spiral8x8/Insert	0,00824 (+/- 0,00037)	0,00682 (+/- 0,00044)	0,00059 (+/- 0,00013)
MMASUS/Insert - Spiral/Insert	0,01297 (+/- 0,00101)	0,00867 (+/- 0,00219)	0,00074 (+/- 0,00026)
MMASUS/Insert - NN 2-Opt/Insert	0,00087 (+/- 0,00001)	0,00086 (+/- 0,00003)	0,00051 (+/- 0,00004)
MMASUS/Insert - NN 2-Opt/NN 2-Opt	0,00149 (+/- 0,00001)	0,00109 (+/- 0,00007)	0,00110 (+/- 0,00018)
MMASUS/US - Spiral8x8/Insert	0,01239 (+/- 0,00202)	0,00631 (+/- 0,00118)	0,00029 (+/- 0,00018)
MMASUS/US - Spiral/Insert	0,01616 (+/- 0,00039)	0,01093 (+/- 0,00248)	0,00025 (+/- 0,00006)
MMASUS/US - NN 2-Opt/Insert	0,00235 (+/- 0,00054)	0,00140 (+/- 0,00004)	0,00038 (+/- 0,00026)
MMASUS/US - NN 2-Opt/NN 2-Opt	0,00286 (+/- 0,00011)	0,00159 (+/- 0,00010)	0,00018 (+/- 0,00007)
Spiral8x8/Insert - Spiral/Insert	0,00682 (+/- 0,00031)	0,00463 (+/- 0,00121)	0,00027 (+/- 0,00016)
Spiral8x8/Insert - NN 2-Opt/Insert	0,00756 (+/- 0,00034)	0,00569 (+/- 0,00108)	0,00019 (+/- 0,00005)
Spiral8x8/Insert - NN 2-Opt/NN 2-Opt	0,00863 (+/- 0,00060)	0,00540 (+/- 0,00119)	0,00038 (+/- 0,00021)
Spiral/Insert - NN 2-Opt/Insert	0,01317 (+/- 0,00050)	0,00853 (+/- 0,00250)	0,00021 (+/- 0,00003)
Spiral/Insert - NN 2-Opt/NN 2-Opt	0,01362 (+/- 0,00089)	0,01002 (+/- 0,00199)	0,00034 (+/- 0,00016)
NN 2-Opt/Insert - NN 2-Opt/NN 2-Opt	0,00131 (+/- 0,00007)	0,00139 (+/- 0,00005)	0,00120 (+/- 0,00056)

6.1.2 Nachweis der erfolgreichen Selektion von Algorithmen

Für den Nachweis der erfolgreichen Selektion von Algorithmen stehen Probleminstanzen zur Verfügung, die für den Spiral/Insert bzw. NN 2-Opt/US-Algorithmus erzeugt wurden und von allen Algorithmen gelöst sind. Für den Nachweis werden ausschließlich Instanzen verwendet, die nicht am Training oder der Evaluation der implementierten Regressionsmodelle beteiligt sind (vgl. Abschnitt 6.1). Diese Instanzen werden in zwei Mengen aufgeteilt. In der ersten Menge (Spiral/Insert-Menge) befinden sich ausschließlich Instanzen erzeugt für den Spiral/Insert-Algorithmus. Abbildung 6.2 zeigt die prozentualen Anteile von Probleminstanzen, für den ein Algorithmus das beste Ergebnis im Vergleich zu allen anderen Algorithmen erzielt. In der zweiten Menge (NN 2-Opt/US-Menge) sind die Probleminstanzen, die für den NN 2-Opt/US-Algorithmus erzeugt und von allen anderen Algorithmen gelöst sind. Die Anteile in der zweiten Menge sind ähnlich verteilt wie in Abbildung 6.2.

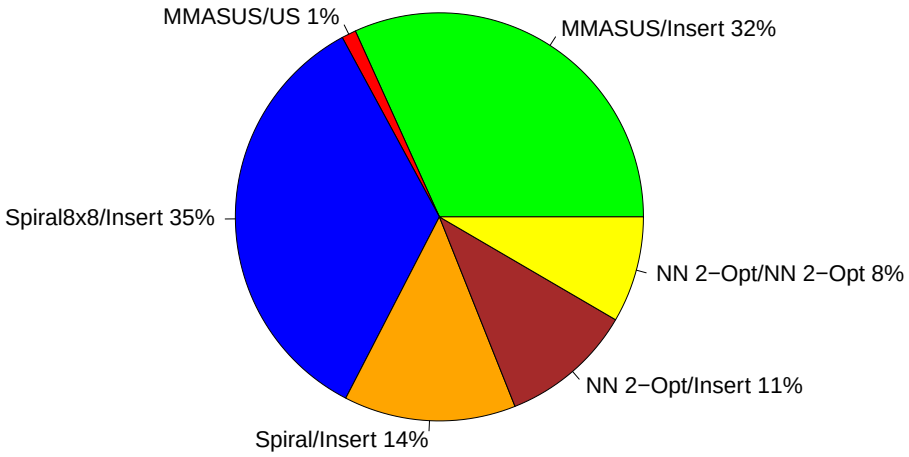

Abbildung 6.2: Anteile von Probleminstanzen, für den ein Algorithmus das beste Ergebnis im Vergleich zu allen anderen Algorithmen erzielt

Die Anteile, abgebildet in Abbildung 6.2, sind denen der gesamten Daten sehr ähnlich (vgl. Abbildung 5.9). Dieser Sachverhalt ist wahrscheinlich auf die gleichmäßige Verteilung der statischen Instanzen und DOD auf die Mengen der Probleminstanzen zurückzuführen (vgl. Abschnitt 6.1). Abbildung 6.2 zeigt, dass jeder Algorithmus für einen signifikanten Anteil von Probleminstanzen die höchste Lösungsqualität aufweist. Damit eignen sich die Mengen von Probleminstanzen für den Nachweis der Selektion von Algorithmen. Ausschließlich der NN 2-Opt/US-Algorithmus wird vollständig überdeckt.

Tabelle 6.3 und Tabelle 6.4 zeigen die mittleren quadratischen Fehler der Regressions- und paarweisen Regressionsmodelle, erarbeitet im vorherigen Abschnitt, angewendet auf die Spiral/Insert-Menge von Probleminstanzen. Die gemessenen Fehler sind ähnlich groß im Vergleich zu den Ergebnissen der Kreuzvalidierung der Modelle (vgl. Tabelle 6.1 und Tabelle 6.2). Die Vorhersagegenauigkeit der Modelle ist demzufolge hoch, was auf ein gutes Ergebnis bei der Selektion von Algorithmen schließen lässt. Gut ist zu erkennen, dass alle Modelle für die unbekannten Probleminstanzen leicht schlechtere Ergebnisse erzielen. Zu beachten ist aber, dass in Tabelle 6.1 und Tabelle 6.2 der Durchschnitt der Fehler aus fünf verschiedenen Evaluierungen mit unterschiedlichen und weniger Probleminstanzen abgebildet sind.

Tabelle 6.3: Ergebnisse der Regressionsmodelle für alle Probleminstanzen (für Spiral/Insert erzeugt und durch alle Algorithmen gelöst)

Algorithmus	Mittlerer quadratische Fehler		
	4 wichtigsten Eigenschaften	8 wichtigsten Eigenschaften	Alle Eigenschaften
MMASUS/Insert	0,00652	0,00079	0,00147
MMASUS/US	0,00900	0,00143	0,00272
Spiral8x8/Insert	0,00964	0,00533	0,00415
Spiral/Insert	0,01935	0,01059	0,00809
NN 2-Opt/Insert	0,00585	0,00099	0,00181
NN 2-Opt/NN 2-Opt	0,00785	0,00129	0,00184
NN 2-Opt/US	0,01015	0,00149	0,00274

Tabelle 6.4: Ergebnisse der paarweisen Regressionsmodelle für alle Probleminstanzen (für Spiral/Insert erzeugt und durch alle Algorithmen gelöst)

Algorithmus	Mittlerer quadratische Fehler		
	4 wichtigsten Eigenschaften	8 wichtigsten Eigenschaften	Alle Eigenschaften
MMASUS/Insert - MMASUS/US	0,00156	0,00111	0,00063
MMASUS/Insert - Spiral8x8/Insert	0,00520	0,00493	0,00078
MMASUS/Insert - Spiral/Insert	0,01210	0,01177	0,00132
MMASUS/Insert - NN 2-Opt/Insert	0,00095	0,00088	0,00068
MMASUS/Insert - NN 2-Opt/NN 2-Opt	0,00091	0,00071	0,00076
MMASUS/US - Spiral8x8/Insert	0,00696	0,00528	0,00023
MMASUS/US - Spiral/Insert	0,01121	0,01355	0,00054
MMASUS/US - NN 2-Opt/Insert	0,00214	0,00144	0,00014
MMASUS/US - NN 2-Opt/NN 2-Opt	0,00197	0,00118	0,00013
Spiral8x8/Insert - Spiral/Insert	0,01201	0,01160	0,00058
Spiral8x8/Insert - NN 2-Opt/Insert	0,00467	0,00400	0,00020
Spiral8x8/Insert - NN 2-Opt/NN 2-Opt	0,00520	0,00393	0,00022
Spiral/Insert - NN 2-Opt/Insert	0,01184	0,01335	0,00051
Spiral/Insert - NN 2-Opt/NN 2-Opt	0,01063	0,01208	0,00062
NN 2-Opt/Insert - NN 2-Opt/NN 2-Opt	0,00076	0,00070	0,00010

Für den Nachweis der erfolgreichen Selektion von Algorithmen werden für alle Probleminstanzen alle Problemeigenschaften bestimmt. Auf der Grundlage der Eigenschaften können durch die Modelle $S_{reg}(f(x))$ und $S_{p-reg}(f(x))$ Algorithmen selektiert werden. Die erzielte Lösungsqualität des selektierten Algorithmus wird für jedes Selektionsmodell aufaddiert. Da alle Lösungen der Algorithmen für die Probleminstanzen

bekannt sind, können die Gesamtkosten für jeden Algorithmus separat ermittelt werden. Ein erfolgreiches Modell $S(f(x))$ muss das Minimum dieser Gesamtkosten unterbieten. Die Selektion von Algorithmen ist nur dann sinnvoll, wenn die erzeugten Gesamtlösungskosten niedriger sind, als die des im Durchschnitt über alle Probleminstanzen am besten agierenden Algorithmus. Aufgrund der bekannten Lösungsqualitäten kann ein Orakel die optimale Selektion der Algorithmen ermitteln. Dieses Optimum stellt die maximal zu erreichende Qualität eines Modells $S(f(x))$ dar. Da die Lösungsqualität mit dem erweiterten Value of Information erfasst wird, entsprechen kleinere Werte einer höheren Qualität (vgl. Unterabschnitt 2.1.6). Alle Ergebnisse der Analysen sind in Tabelle 6.5 abgebildet.

Tabelle 6.5: Ergebnisse der Algorithmenselektion für Probleminstanzen (für Spiral/Insert erzeugt und durch alle Algorithmen gelöst)

Algorithmus	Summe der *Value of information*	∅ Rang
MMASUS/Insert	24.270,97	2,38
MMASUS/US	28.770,63	5,40
Spiral8x8/Insert	25.000,49	3,23
Spiral/Insert	27.429,83	4,23
NN 2-Opt/Insert	25.106,53	3,16
NN 2-Opt/NN 2-Opt	25.380,42	3,51
NN 2-Opt/US	29.576,32	6,06
Orakel (Minimum)	22.449,42	1
Zufällige Auswahl	26.018,71	3,47
Paarweise RM mit 4 Eigenschaften	24.377,67	2,46
Paarweise RM mit 8 Eigenschaften	24.358,60	2,45
Paarweise RM mit allen Eigenschaften	24.367,24	2,45
RM mit 4 wichtigsten Eigenschaften	24.371,49	2,63
RM mit 8 wichtigsten Eigenschaften	**23.339,11**	**2,02**
RM mit allen Eigenschaften	**23.762,94**	**2,33**

Im ersten Teil der Tabelle 6.5 sind die Summen für die Algorithmen erfasst. Wenn, zum Beispiel, alle Probleminstanzen mit dem MMASUS/Insert-Algorithmus gelöst werden, wird eine Lösungsqualität von 24.270,97 erzielt. Der MMASUS/Insert-Algorithmus erzeugt für alle Instanzen die niedrigsten Kosten und ist 8,11% vom möglichen Optimum (Orakel) entfernt. Ein erfolgreiches Modell $S(f(X))$ muss geringere Kosten erzeugen als MMASUS/Insert. Der Algorithmus, der die höchsten Kosten verursacht, ist der NN 2-Opt/US-Algorithmus. Dieser ist 31,74% vom möglichen Optimum entfernt. Werden Algorithmen zufällig ausgewählt kann eine Qualität von 26.018,71 erzielt werden. Im dritten Teil der Tabelle sind die Summen für die erarbeiteten Modelle erfasst. Das Modell $S_{p-reg}(f(x))$ (paarweise RM) kann mit keiner untersuchten Anzahl von Problemeigenschaften Algorithmen so selektieren, dass über alle Probleminstanzen ein

besseres Ergebnis im Vergleich zu MMASUS/Insert erzielt wird. Das beste Ergebnis wird hier für die 8 wichtigsten Problemeigenschaften erzeugt und ist 8,50% vom Optimum entfernt. Die Modelle $S_{p-reg}(f(x))$ erreichen ähnlich hohe Lösungsqualitäten wie der beste Algorithmus MMASUS/Insert. Das Modell $S_{reg}(f(x))$ (RM) erreicht mithilfe der acht wichtigsten und allen Eigenschaften eine deutlich höhere Lösungsqualität als der beste Algorithmus. Der Abstand zum Optimum beträgt nur 3,96%. Die Selektion von Algorithmen mit den wichtigsten acht Problemeigenschaften mit $S_{reg}(f(x))$ ist 4,15% besser als der erfolgreichste Algorithmus. Auch die erreichte Lösungsqualität durch die Selektion auf Basis aller Eigenschaften ist deutlich besser im Vergleich zur Qualität, erreicht durch den besten Algorithmus. Damit ist grundsätzlich nachgewiesen, dass auf der Grundlage der in dieser Arbeit eingeführten Problemeigenschaften eine automatisierte Selektion von Algorithmen für Online-Optimierungsprobleme am Beispiel des DVRP möglich ist. Werden nur die vier wichtigsten Problemeigenschaften für die Selektion herangezogen, kann das Modell $S_{reg}(f(x))$ die Algorithmen nicht erfolgreich selektieren. Es scheint, dass nur die wichtigsten vier Eigenschaften nicht ausreichend sind, um die Beziehungen zwischen Probleminstanz und Performance eines Algorithmus hinreichend zu beschreiben. Mit dieser Erkenntnis lässt sich der erreichte Fehler der Regressionsmodelle, erfasst in Tabelle 6.1 und Tabelle 6.2, bewerten. Die Vorhersagegenauigkeit von Modellen mit einem Fehler $> 0{,}01$ ist für die erfolgreiche Selektion von Algorithmen nicht ausreichend. Eine ausführliche Diskussion über das nicht erfolgreiche Abschneiden der paarweisen Regressionsmodelle erfolgt unter anderem in Abschnitt 6.2.

Neben den aufsummierten Lösungskosten ist in Tabelle 6.5 auch der durchschnittliche Rang für jeden Algorithmus und jedes Modell angegeben. Der Rang stellt das Ergebnis eines Algorithmus in Bezug zu den Ergebnissen aller anderen Algorithmen dar. Ein Rang von 1,0 für eine Probleminstanz kennzeichnet einen Algorithmus als denjenigen, der die niedrigsten Lösungskosten für die Instanz generiert. Einen Rang von 7,0 erreicht der Algorithmus, der die höchsten Lösungskosten erzeugt. Der Rang berücksichtigt ausschließlich die betrachteten Algorithmen, nicht die Selektionsmodelle. Das Orakel hat einen durchschnittlichen Rang von 1,0. Hier wird immer der Algorithmus selektiert, der die niedrigsten Kosten verursacht. Werden zufällig Algorithmen ausgewählt ist der Erwartungswert des Ranges 3,5. Bei den durchgeführten Experimenten ist der durchschnittliche Rang 3,47 für die zufällige Auswahl eines Algorithmus. Die durchschnittlichen Ränge für alle Modelle $S(f(x))$ bewegen sich im Bereich 2,0 bis 2,6 und weisen auf eine solide Selektion von Algorithmen hin. Keines der Modelle weist einen schlechteren durchschnittlichen Rang auf als der zweitbeste Algorithmus des Portfolios (NN 2-Opt/Insert). Die Modelle $S_{reg}(f(x))$ für die wichtigsten acht und alle Problemeigenschaften unterbieten den Rang des durchschnittlich erfolgreichsten Algorithmus MMASUS/Insert. Abbildung 6.3 zeigt die Rangverteilung für alle Algorithmen und Modelle in Form von Box-Plots. Sowohl für die erfolgreichen Modelle $S_{reg}(f(x))$ als auch für die Modelle $S_{p-reg}(f(x))$ zeigt sich, dass 75% der getroffenen Entscheidungen einen Rang von 1,0 bis 3,0 aufweisen. Es zeigt sich aber auch, dass durch die Modelle $S_{reg}(f(x))$ in Ausnahmefällen auch der schlechtest mögliche Algorithmus selektiert

wird. Ein schlechtest mögliches Ergebnis bei der Auswahl eines Algorithmus für eine Probleminstanz kann also durch das Einsetzen der Selektionsmodelle nicht vermieden werden. Bei der Interpretation der Box-Plots ist zu beachten, dass der Algorithmus NN 2-Opt/US durch die Modelle $S_{p-reg}(f(x))$ (paarw. RM) nicht selektiert werden kann.

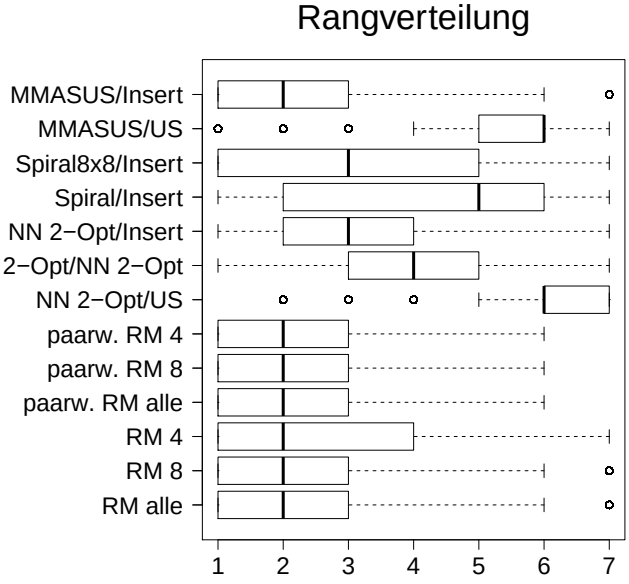

Abbildung 6.3: Box-Plots der Verteilung der Ränge der Algorithmen für Probleminstanzen (für NN 2-Opt/US erzeugt und durch alle Algorithmen gelöst wurden)

Abbildung 6.4 zeigt die Verteilung der Ränge des erfolgreichsten $S_{reg}(f(x))$ Modells (berücksichtigt die acht wichtigsten Eigenschaften) und des Algorithmus MMASUS/Insert. Wie zu erwarten ist, erzielt das Selektionsmodell signifikant häufiger den besten Rang im Vergleich zu MMASUS/Insert. Abbildung 6.4 visualisiert aber auch, dass das Modell leicht häufiger auch den schlechtesten Algorithmus für die Bearbeitung der Probleminstanz auswählt. Sollen in einer Problemstellung explizit schlechteste Ergebnisse vermieden werden, deuten die Histogramme darauf hin, dass der MMASUS/Insert-Algorithmus dem Selektionsmodell $S_{reg}(f(x))$ bei einer solchen Zielstellung zu bevorzugen ist. Bei der detaillierten Analyse der Differenzen zwischen den erzielten Kosten und den minimal möglichen Kosten bei einem Rang größer 5,0, zeigt sich jedoch, dass sich die Differenzen nur minimal voneinander unterscheiden (vgl. Abbildung 6.5). 775 mal erreicht der Algorithmus MMASUS/Insert und 1.306 mal das Modell $S_{reg}(f(x))$ einen Rang größer 5,0. Die Differenzen zum besten Algorithmus summieren sich für den MMASUS/Insert-Algorithmus auf 61 und für das Modell auf 97. Im Durchschnitt ist die Distanz der schlechtesten Lösung, erzeugt durch das Selektionsmodell, zum Optimum kleiner im Vergleich zu der Distanz erzeugt durch MMASUS/Insert (vgl. Abbildung 6.5). Der Einfluss der 1.306 Selektionen, bei denen ein besonders schlechter Algorithmus ausgewählt wird, beläuft sich lediglich auf 0,43%-Punkte am optimalen

Gesamtergebnis. So zeigt sich, dass die Ergebnisse, für die der schlechteste Algorithmus ausgewählt wird, keinen signifikanten negativen Einfluss auf das Gesamtergebnis haben und dass das schlechteste Ergebnis, ausgewählt vom Selektionsmodell im Durchschnitt besser ist, als das schlechteste Ergebnis, ausgewählt von MMASUS/Insert. Das bedeutet, dass die Ergebnisse der Algorithmen sehr nah beieinander liegen, wenn einer der schlechtesten Algorithmen ausgewählt wird. Aus diesem Grund muss der Gebrauch des Selektionsmodells $S_{reg}(f(x))$ auch bei einer Zielstellung, bei der das Vermeiden von schlechtesten Ergebnissen angestrebt wird, empfohlen werden. Die Rangverteilungen aller betrachteten Ansätze sind in Abschnitt B.2 zu finden.

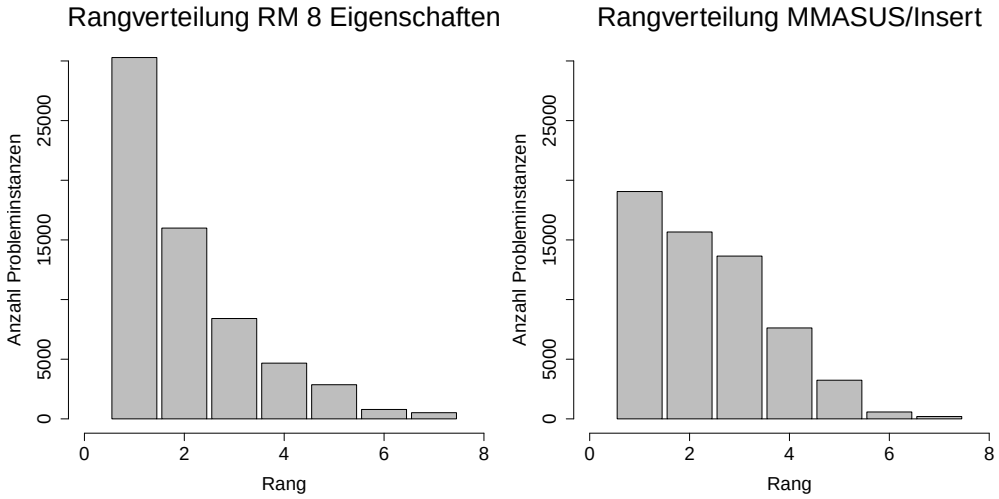

Abbildung 6.4: Histogramme der Verteilung der Ränge der Algorithmen für Probleminstanzen (für NN 2-Opt/US erzeugt und durch alle Algorithmen gelöst)

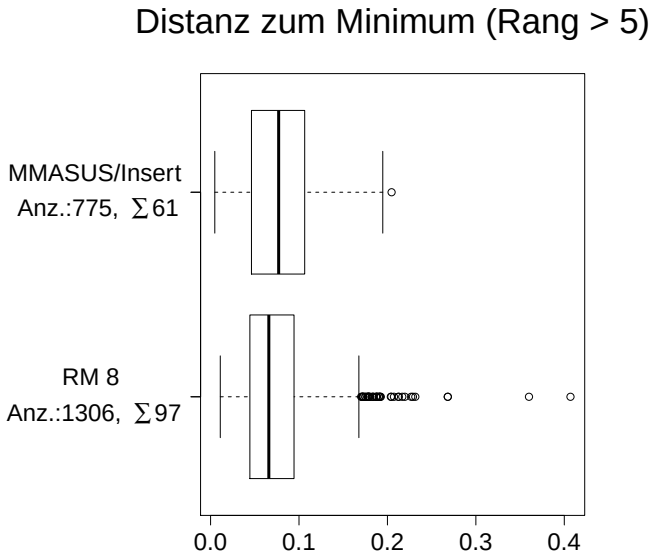

Abbildung 6.5: Box-Plots der Verteilung der Distanzen zum Minimum bei einem Rang größer 5,0

Zusammenfassend zeigt sich, dass das erarbeitete Modell $S_{reg}(f(x))$ für die acht wichtigsten und alle Problemeigenschaften zuverlässig die besten Algorithmen für Probleminstanzen selektiert, und im Durchschnitt erfolgreicher als die Anwendung des besten Algorithmus ist. Das Modell, das alle Problemeigenschaften nutzt, erreicht leicht schlechtere Ergebnisse bei der Selektion von Algorithmen, trotz leicht höherer Vorhersagegenauigkeit für die Trainingsdaten (vgl. Tabelle 6.1 in Unterabschnitt 6.1.1). Für die unbekannten Daten hingegen ist die Genauigkeit der Modelle, die alle Eigenschaften betrachten, geringer im Vergleich zu denen, die nur die acht wichtigsten Problemeigenschaften nutzen (vgl. Tabelle 6.3). Wird für Instanzen der schlechteste Algorithmus ausgewählt, unterscheiden sich die Lösungsqualitäten der Algorithmen nur geringfügig, was auf für die Algorithmen ähnlich schwer zu lösende Probleminstanzen schließen lässt. Eine erfolgreiche Selektion auch bei diesen Instanzen trägt nur wenig zur Verbesserung der Gesamtlösungsqualität bei. Die erarbeiteten Selektionsmodelle $S_{p-reg}(f(x))$ zeigen sich hingegen nicht erfolgreich bei der Auswahl von Algorithmen für die gegebenen Probleminstanzen, trotz höherem Vertrauen in die Entscheidungskriterien durch das Nutzen von mehreren Modellen (vgl. Unterabschnitt 6.1.1). Die in Xu et al. (2011) eingeführten, erfolgreichen paarweisen Klassifikationsmodelle werden mit Probleminstanzen trainiert, bei denen der Abstand der Lösungsqualitäten der beteiligten Algorithmen besonders hoch ist. Dieser Umstand ist beim Training der Modelle $S_{p-reg}(f(x))$ nicht berücksichtigt und trägt unter Umständen zum eher schlechteren Abschneiden bei. Für die zweite Menge (NN 2-Opt/US-Menge) der für das Validieren der Modelle bereit stehenden Probleminstanzen, kann dennoch gezeigt werden, dass das Selektionsmodell

$S_{p-reg}(f(x))$ erfolgreich ist (vgl. Tabelle 6.6). Hier liegt die Verbesserung durch die Selektion von Algorithmen im Vergleich zum besten Algorithmus aber lediglich bei rund 0,07%-Punkten. Auch für die zweite Menge sind die Modelle $S_{reg}(f(x))$, die acht bzw. alle Problemeigenschaften berücksichtigen, sehr erfolgreich.

Tabelle 6.6: Ergebnisse der Algorithmenselektion für Probleminstanzen (für NN 2-Opt/US erzeugt und durch alle Algorithmen gelöst)

Algorithmus	Summe der *Value of information*	∅ Rang
MMASUS/Insert	26.624,16	2,38
MMASUS/US	31.726,40	5,32
Spiral8x8/Insert	28.605,87	3,45
Spiral/Insert	31.023,50	4,22
NN 2-Opt/Insert	27.601,61	3,15
NN 2-Opt/NN 2-Opt	27.767,61	3,47
NN 2-Opt/US	32.877,12	5,98
Orakel (Minimum)	24.573,10	1
Zufällige Auswahl	28.879,13	3,64
Paarweise RM mit 4 Eigenschaften	26.749,89	2,47
Paarweise RM mit 8 Eigenschaften	**26.606,98**	**2,41**
Paarweise RM mit allen Eigenschaften	26.764,27	2,49
RM mit 4 wichtigsten Eigenschaften	26.893,26	2,68
RM mit 8 wichtigsten Eigenschaften	**26.103,84**	**2,32**
RM mit allen Eigenschaften	**26.164,38**	**2,34**

6.1.3 Evaluation der Modelle mit unbekannte Instanzen

Neben dem Nachweis der erfolgreichen Selektion von Algorithmen mit dynamischen Probleminstanzen, die aus den statischen Problemstellungen aus Abbildung 5.2 erzeugt sind, soll das erfolgreichste Selektionsmodell zusätzlich mithilfe dynamischer Instanzen evaluiert werden, die aus unbekannten Probleminstanzen erzeugt sind. Unbekannte Instanzen sind nicht Teil der in Unterabschnitt 5.1.2 beschriebenen Experimente. Dynamische Problemstellungen, generiert aus unbekannten statischen Instanzen, werden als unbekannte dynamische Probleminstanzen bezeichnet. Bekannte statische Problemstellungen sind Teil der in Unterabschnitt 5.1.2 beschriebenen Experimente. Bekannte dynamische Instanzen werden aus bekannten statischen Problemstellungen generiert, sind aber weder am Training noch an der Evaluation der Regressionsmodelle beteiligt. Alle verwendeten unbekannten statischen Problemstellungen sind in Abbildung 6.6 abgebildeten.

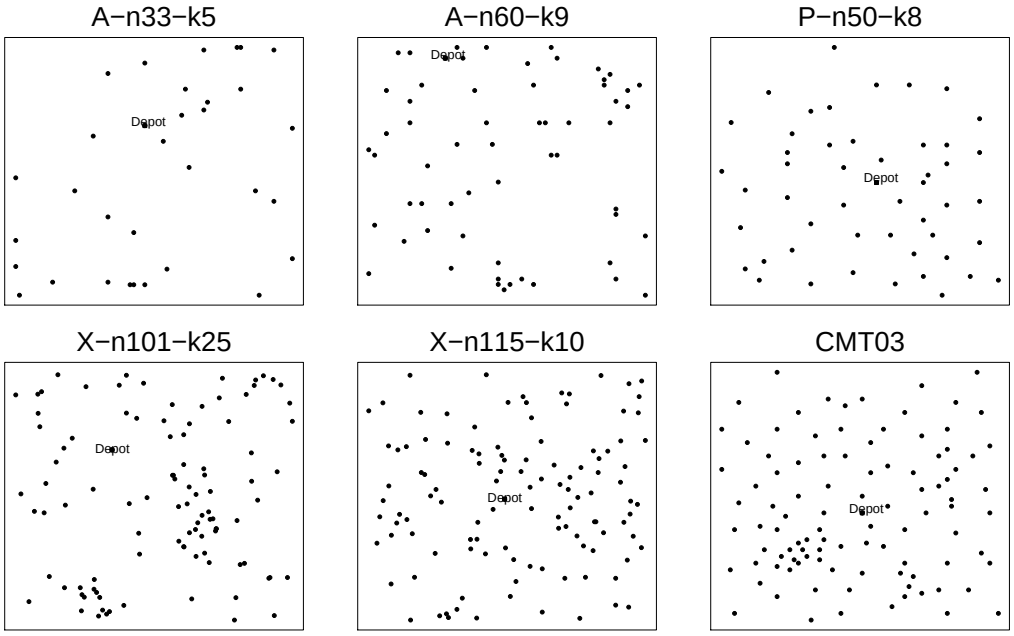

Abbildung 6.6: Unbekannte statischen Probleminstanzen für die Evaluation der Selektionsmodelle

Die Auswahl der unbekannten statischen Instanzen erfolgt zufällig. Die Instanzen A-n33-k5, A-n60-k9 und P-n50-k8, eingeführt von Christiansen und Lysgaard (2007), umfassen 33, 60, bzw. 50 Kundenanfragen. Aufgrund der Anzahl und der Verteilung der Anfragen sind die Instanzen den bekannten Probleminstanzen CMT01 und CMT02 ähnlich (vgl. Abbildung 5.2). A-n33-k5 und A-n60-k9 weisen vereinzelt größere Flächen ohne Anfragen auf und haben aus diesem Grund ebenfalls Elemente der bekannten Instanzen CMT11TM50R, CMT12TM50R und Golden_17 (vgl. Abbildung 5.2). X-n101-k25 und X-n115-k10 bestehen aus 101 bzw. 115 Anfragen und resultieren aus der Arbeit von Uchoa et al. (2017). Unter den zum Training der Netze verwendeten Probleminstanzen finden sich keine Instanzen mit gleichverteilten Anfragen, die eine so hohe Anzahl von Anfragen modellieren. Die Instanz X-n101-k25 weist vereinzelte Cluster mit zusätzlich im Problemraum gleichmäßig verteilten Anfragen auf, ähnelt also den Instanzen CMT11TM50R und CMT12TM50R (vgl. Abbildung 5.2). Die Probleminstanz CMT03 stammt, wie viele der am Training beteiligten Instanzen, von Christofides (1976) und beschreibt 101 gleichverteilte Kundenanfragen. Aufgrund der Anzahl und der Verteilung der Anfragen ist die Instanz der bekannten Instanz CMT02 ähnlich. Die Depots aller unbekannten Instanzen sind überwiegend in der Nähe des Zentrums platziert. Eine Ausnahme bildet A-n60-k9, hier ist das Depot nordwestlich des Zentrums am Rand der Problemstellung angesiedelt. Die Depots der bekannten Instanzen sind ebenfalls überwiegend im Zentrum angesiedelt. Vereinzelt (CMT11, CMT11TM50R und CMT12TM50R) weicht die Lage des Depots aber auch ab (vgl. Abbildung 5.2). Keine der unbekannten Probleminstanzen weist unberücksichtigte Eigenschaften auf,

aber auch keine unbekannte Instanz weist ausschließlich Merkmale auf, die nur einer bekannten Probleminstanz zugeordnet werden können. Die betrachteten unbekannten Instanzen eignen sich aus diesem Grund, um die Auswahl der an den Experimenten beteiligten bekannten Instanzen (vgl. Unterabschnitt 5.1.2) zu bewerten. Kann das Selektionsmodell auch für die unbekannten dynamischen Problemstellungen Algorithmen erfolgreich selektieren, bildet die Menge der betrachteten bekannten Instanzen eine solide Grundlage für das Erstellen der Regressionsmodelle. Mit weiteren Einflussfaktoren auf den Erfolg der Selektionsmodelle setzt sich unter anderem Abschnitt 6.2 auseinander.

Für alle Problemstellungen aus Abbildung 6.6 werden 1.000 dynamische Instanzen zufällig mit einem DOD von 0,3 erzeugt. Zusätzlich werden zielgerichtete Probleminstanzen, ebenfalls mit einem DOD von 0,3, generiert, die die Flächen-Eigenschaft (FE) minimieren bzw. maximieren. Bei dem zielgerichteten Generieren der 2.000 zusätzlichen Instanzen kommt die Implementierung, vorgestellt in Unterabschnitt 4.1.1 und Unterabschnitt 5.1.1, zum Einsatz. Instanzen mit einer besonders kleinen bzw. großen Ausprägung der Flächen-Eigenschaft werden generiert, da die Eigenschaft einen wesentlichen Einfluss auf die Lösungsqualität hat (vgl. Unterabschnitt 5.2.1). Alle 18.000 Problemstellungen werden mit einem Vertreter je Algorithmenklasse (vgl. Unterabschnitt 5.1.3) gelöst. Es werden die folgenden Vertreter berücksichtigt: MMASUS/Insert, MMASUS/US, Spiral8x8/Insert und Spiral/Insert. Für die Evaluation der Probleminstanzen müssen 7,2 Millionen Simulationen durchgeführt werden (vgl. Unterabschnitt 5.1.2). Erzeugen die Algorithmen für Instanzen nicht signifikant unterschiedliche Lösungsqualitäten (die Varianz der Lösungen ist kleiner 0,001) werden diese für die Auswahl von Algorithmen nicht betrachtet. Die Auswahl von Algorithmen für sehr ähnliche Problemstellungen gestaltet sich als schwierig und hat keinen signifikanten Einfluss auf das Gesamtergebnis (vgl. Unterabschnitt 6.1.2). Die Verteilung der Anteile der betrachteten 10.300 Probleminstanzen, für den ein Algorithmus das beste Ergebnis im Vergleich zu allen anderen Algorithmen erzielt, ist in Abbildung 6.7 abgebildet. Jeder Algorithmus löst einen signifikanten Anteil von Probleminstanzen am besten. Damit eignen sich die generierten Instanzen für eine Selektion von Algorithmen.

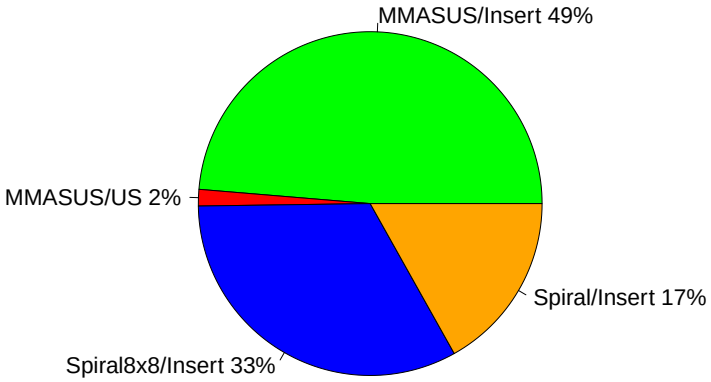

Abbildung 6.7: Anteil von Probleminstanzen, für den ein Algorithmus das beste Ergebnis im Vergleich zu allen anderen Algorithmen erzielt (für unbekannte Probleminstanzen)

Tabelle 6.7 zeigt den mittleren quadratischen Fehler erzeugt durch die Regressionsmodelle bei der Anwendung auf die unbekannten Probleminstanzen. Zusätzlich sind die Fehler der Modelle aus Unterabschnitt 6.1.2 gelistet. Die Unterschiede der Fehler sind für die Algorithmen signifikant. Die Vorhersagegenauigkeit für die unbekannten Instanzen ist wesentlich geringer, der Fehler größer als 0,01. Die Größe des Fehlers lässt vermuten, dass eine erfolgreiche Selektion von Algorithmen nicht möglich ist. Ausschließlich das Modell für Spiral/Insert erzeugt einen ähnlich großen Fehler für beide Gruppen von Probleminstanzen. Die Ergebnisse deuten darauf hin, dass die Regressionsmodelle auf die dynamischen Instanzen, erzeugt aus den statischen Instanzen abgebildet in Abbildung 5.2, spezialisiert sind. Eine intensive Diskussion über Einflussfaktoren auf die Vorhersagegenauigkeit findet unter anderem in Abschnitt 6.2 statt.

Tabelle 6.7: Ergebnisse der Regressionsmodelle für alle unbekannten Probleminstanzen im Vergleich zu den Ergebnissen für Instanzen beschrieben in Unterabschnitt 6.1.2

Algorithmus	Mittlerer quadratische Fehler	
	verwandte Instanzen	unbkannte Instanzen
MMASUS/Insert	0,00079	0,03515
MMASUS/US	0,00143	0,03121
Spiral8x8/Insert	0,00533	0,02268
Spiral/Insert	**0,01059**	**0,01112**

Trotz der geringeren Vorhersagegenauigkeit ist das Selektionsmodell erfolgreich bei der Auswahl von Algorithmen für die unbekannten Probleminstanzen. Tabelle 6.8 zeigt im ersten Teil die Summen aller erzeugten Kosten für die Probleminstanzen für jeden betrachteten Algorithmus. Die Gesamtkosten der optimalen Selektion der Algorithmen und das Resultat einer zufälligen Auswahl sind im zweiten Teil der Tabelle dargestellt. In

der letzten Zeile ist das Ergebnis des Selektionsmodells $S_{reg}(f(x))$ unter Berücksichtigung der wichtigsten acht Problemeigenschaften dargestellt. Das Selektionsmodell erzeugt nur 6,1% höhere Kosten als eine optimale Auswahl und ist damit um 0,8%-Punkte besser im Vergleich zum besten Algorithmus MMASUS/Insert. Der Performance-Gewinn durch die automatisierte Selektion von Algorithmen ist geringer ausgeprägt im Vergleich zu den Probleminstanzen aus Unterabschnitt 6.1.2. Dieser Sachverhalt ist auf die schlechtere Vorhersagegenauigkeit der Regressionsmodelle, verwendet in $S_{reg}(f(x))$, zurückzuführen. Es zeigt sich, dass auch Fehler größer als 0,01 eine erfolgreiche Selektion von Algorithmen erlauben. Zu berücksichtigen ist aber, dass Probleminstanzen mit nur geringer Abweichung der Lösungsqualitäten nicht berücksichtigt werden. Eine erfolgreiche Selektion zwischen Algorithmen für sehr ähnliche Probleminstanzen und sehr ähnliche Lösungsqualitäten ist mit Regressionsmodellen mit einem Fehler größer 0,01 nicht möglich.

Tabelle 6.8: Ergebnisse der Algorithmenselektion für alle unbekannten Probleminstanzen

Algorithmus	Summe der *Value of information*	∅ Rang
MMASUS/Insert	4.176,38	1,93
MMASUS/US	4.794,92	3,45
Spiral8x8/Insert	4.273,36	2,06
Spiral/Insert	4.407,11	2,54
Orakel (Minimum)	3.904,27	1
Zufällige Auswahl	4.419,00	2,44
RM mit 8 wichtigsten Eigenschaften	**4.143,28**	**1,87**

Tabelle 6.9 zeigt die Ergebnisse für die Algorithmen und das Selektionsmodell für eine Teilmenge der unbekannten Probleminstanzen. Die Menge beschränkt sich auf dynamische Instanzen generiert aus der Problemstellung P-n50-k8. Hier zeigt sich ein ähnliches Bild wie für alle betrachteten Instanzen. MMASUS/Insert erreicht für alle Problemstellungen die niedrigsten Gesamtkosten und ist 8,6% vom Optimum entfernt. Das Modell $S_{reg}(f(x))$ ist nur 6,3% vom Optimum entfernt, ist also um 2,2%-Punkte besser als der erfolgreichste Algorithmus. Für die Probleminstanzen A-n33-k5 und A-n60-k9 verhält sich das Selektionsmodell ähnlich.

Werden ausschließlich die Ergebnisse für die Probleminstanz X-n101-k25 betrachtet zeigt sich, dass das Modell $S_{reg}(f(x))$ keinen Vorteil gegenüber dem Algorithmus Spiral8x8/Insert, der die niedrigsten Gesamtkosten erzeugt, bietet. Das Modell wählt für alle dynamischen Probleminstanzen den Algorithmus Spiral8x8/Insert aus und erreicht damit ebenfalls einen Abstand von nur rund 4,5% zum Optimum (vgl. Tabelle 6.10). Zusammenfassend zeigt sich, dass das Selektionsmodell für die unterschiedlichen unbekannten Probleminstanzen unterschiedlich gute Ergebnisse generiert. Es zeigt sich aber, dass das Selektionsmodell im Durchschnitt auch für die unbekannten dynamischen Problemstellungen Algorithmen erfolgreich selektiert. Die Menge der betrachteten

Tabelle 6.9: Ergebnisse der Algorithmenselektion für Probleminstanzen mit einem DOD von 0,3 basierend auf P-n50-k8

Algorithmus	Summe der *Value of information*	∅ Rang
MMASUS/Insert	292,36	2,01
MMASUS/US	328,21	3,30
Spiral8x8/Insert	296,07	2,09
Spiral/Insert	313,33	2,58
Orakel (Minimum)	269,23	1
Zufällige Auswahl	313,55	2,46
RM mit 8 wichtigsten Eigenschaften	**286,28**	**1,79**

bekannten Instanzen ist demzufolge eine solide Grundlage für die erstellten Regressionsmodelle. Auch an dieser Stelle wird auf die umfangreiche Diskussion über das Abschneiden der Selektionsmodelle in Abschnitt 6.2 verwiesen.

Tabelle 6.10: Ergebnisse der Algorithmenselektion für Probleminstanzen mit einem DOD von 0,3 basierend auf X-n101-k25

Algorithmus	Summe der *Value of information*	∅ Rang
MMASUS/Insert	1.207,22	1,91
MMASUS/US	1.405,07	3,75
Spiral8x8/Insert	**1.206,03**	**1,68**
Spiral/Insert	1.258,15	2,65
Orakel (Minimum)	1.153,81	1
Zufällige Auswahl	1.257,10	2,41
RM mit 8 wichtigsten Eigenschaften	**1.206,03**	**1,68**

Das Training der Regressionsmodelle basiert auf Probleminstanzen mit einem DOD $\in \{0{,}2; 0{,}3; 0{,}5\}$ (vgl. Unterabschnitt 5.1.2). Alle bisher verwendeten unbekannten dynamischen Instanzen weisen einen DOD von 0,3 auf. Mit den nachfolgenden Experimenten soll evaluiert werden, ob die Modelle auch die Lösungsqualität für Problemstellungen mit unbekannter Dynamik voraussagen können. Dazu werden dynamische Instanzen auf der Grundlage von bekannten (CMT01 und CMT02) und unbekannten (A-n33-k5, A-n60-k9 und P-n50-k8) statischen Instanzen mit einem DOD von 0,4 generiert. Tabelle 6.11 vergleicht den mittleren quadratischen Fehler der Regressionsmodelle für bekannte dynamische Instanzen mit einem $DOD \in \{0{,}2; 0{,}3; 0{,}5\}$ und einem $DOD \in 0{,}4$.

Die Ergebnisse aus Tabelle 6.11 zeigen, dass die Vorhersagegenauigkeit für Problemstellungen mit einem DOD von 0,4 größer ist, als die für die bekannten Instanzen. Zu berücksichtigen ist, dass die Menge der bekannten Instanzen mit einem

Tabelle 6.11: Ergebnisse der Regressionsmodelle für die bekannten Probleminstanzen CMT01 und CMT02 mit einem unbekannten DOD von 0,4 im Vergleich zu den Ergebnissen für Instanzen beschrieben in Unterabschnitt 6.1.2

Algorithmus	Mittlerer quadratische Fehler	
	bekannte Instanzen $DOD \in 0{,}2; 0{,}3; 0{,}5$	bekannte Instanzen $DOD \in 0{,}4$
MMASUS/Insert	0,00079	0,00063
MMASUS/US	0,00143	0,00067
Spiral8x8/Insert	0,00533	0,0012
Spiral/Insert	0,01059	0,00170

$DOD \in \{0{,}2; 0{,}3; 0{,}5\}$ alle bekannten dynamischen Probleminstanzen enthält, also auch diejenigen, für die die Beziehungen zwischen Problemeigenschaften und Lösungsqualität eventuell schlechter zu modellieren sind. Die hohe Qualität der Vorhersagegenauigkeit lässt auf eine erfolgreiche Selektion von Algorithmen schließen. Tabelle 6.12 zeigt die Ergebnisse der automatisierten Auswahl von Algorithmen. Das Selektionsmodell $S_{reg}(f(x))$ ist erfolgreicher als alle betrachteten Algorithmen und 7,9% vom möglichen Optimum entfernt. Damit ist das Modell um 0,24%-Punkte erfolgreicher, als der im Durchschnitt erfolgreichste Algorithmus (Spiral8x8/Insert). Trotz der hohen Vorhersagegenauigkeit der Regressionsmodelle ist das Selektionsmodell nur geringfügig besser im Vergleich zum erfolgreichsten Algorithmus. Grundsätzlich zeigt sich aber, dass auch für dynamische Problemstellungen, mit einer im Training nicht berücksichtigten Dynamik, erfolgreich Algorithmen selektiert werden können. Der Grund dafür liegt in den betrachteten Parametern bei der Generierung von Daten als Trainingsgrundlage für die Regressionsmodelle (vgl. Unterabschnitt 5.1.2).

Tabelle 6.12: Ergebnisse der Algorithmenselektion für Probleminstanzen mit einem DOD von 0,4 basierend auf CMT01 und CMT02

Algorithmus	Summe der *Value of information*	∅ Rang
MMASUS/Insert	639,52	2,18
MMASUS/US	737,62	3,69
Spiral8x8/Insert	632,88	2,03
Spiral/Insert	635,43	2,07
Orakel (Minimum)	585,10	1
Zufällige Auswahl	666,01	2,46
RM mit 8 wichtigsten Eigenschaften	**631,42**	**2,01**

Tabelle 6.13 zeigt die Ergebnisse der Auswahl von Algorithmen für die unbekannten dynamischen Instanzen mit einem DOD von 0,4. Diese basieren auf den Instanzen A-n33-k5, A-n60-k9 und P-n50-k8 (vgl. Abbildung 6.6). Es zeigt sich, dass für die

ausgewählten unbekannten Instanzen, mit im Training nicht berücksichtigter Dynamik, das Selektionsmodell nicht erfolgreich ist. Das Ergebnis des besten Algorithmus, der 7,9% vom Optimum entfernt ist, wird um 0,1%-Punkte verfehlt. Bei unbekannten Instanzen mit unbekannter Dynamik zeigen sich die aktuellen Grenzen der Regressionsmodelle. Gründe für das Abschneiden werden im folgenden Abschnitt 6.2 intensiv diskutiert.

Tabelle 6.13: Ergebnisse der Algorithmenselektion für Probleminstanzen mit einem DOD von 0,4 basierend auf A-n33-k5, A-n60-k9 und P-n50-k8

Algorithmus	Summe der *Value of information*	∅ Rang
MMASUS/Insert	2.760,57	2,08
MMASUS/US	3.177,05	3,68
Spiral8x8/Insert	**2.757,73**	**1,98**
Spiral/Insert	2.788,38	2,24
Orakel (Minimum)	2.553,88	1
Zufällige Auswahl	2.892,88	2,49
RM mit 8 wichtigsten Eigenschaften	2.759,15	2,05

6.2 Zusammenfassung und Diskussion

In Kapitel 6 werden zwei verschiedene Ansätze eines Selektionsmodells $S(f(x))$ vorgestellt. Für jeden Ansatz werden Regressionsmodelle implementiert, die eine unterschiedliche Anzahl von Problemeigenschaften betrachten. In Experimenten wird demonstriert, dass mit den implementierten Modellen eine erfolgreiche Selektion von Algorithmen auf der Basis der in dieser Arbeit vorgestellten acht wichtigsten Problemeigenschaften möglich ist. Zudem wird gezeigt, dass die geschaffene Datenbasis für das Trainieren der Modelle geeignet ist, um auch für gänzlich unbekannte Probleminstanzen erfolgreich Algorithmen zu selektieren. Auch für Problemstellungen mit unbekannter Dynamik sind die Modelle eingeschränkt erfolgreich.

Bei der erfolgreichen Selektion von Algorithmen in Abschnitt 6.2 zeigt sich, dass das Selektionsmodell $S_{p-reg}(f(x))$ schlechter abschneidet, als das Modell $S_{reg}(f(x))$. $S_{p-reg}(f(x))$ basiert auf paarweisen Regressionmodellen, die die Beziehungen zwischen den Problemeigenschaften und der Differenz zweier Algorithmen modellieren. Ein solches Modell sagt nicht die erwartete Lösungsqualität eines Algorithmus voraus, sondern trifft eine Entscheidung darüber, welcher von zwei Algorithmen das Problem voraussichtlich erfolgreicher löst. Einen Algorithmus wählt $S_{p-reg}(f(x))$ demzufolge basierend auf verschiedenen Regressionsmodellen aus. In Wagner et al. (2018) werden paarweise Klassifikationsmodelle als sehr erfolgreich bei der Selektion von Algorithmen beschrieben (vgl. Unterabschnitt 2.2.4). Die Modelle werden mit Probleminstanzen trainiert, für die Algorithmen besonders abweichende Ergebnisse voneinander erzielen. Für das Training

der paarweisen Regressionsmodelle, mit deren Hilfe $S_{p-reg}(f(x))$ Algorithmen selektiert, stehen Problemstellungen zur Verfügung, die ein ausgewählter Algorithmus besonders gut bzw. schlecht löst (vgl. Unterabschnitt 5.1.1). Die Abweichungen der Lösungsqualitäten der Algorithmen können hier ganz unterschiedlich sein. Dieser Umstand scheint den Erfolg von $S_{p-reg}(f(x))$ erheblich zu beeinflussen. Die Generierung von geeigneten Probleminstanzen kann ähnlich erfolgen wie in Abschnitt 5.1 beschrieben. Ein Probleminstanzgenerator navigiert dann nicht anhand der Lösungsqualität eines einzelnen Algorithmus durch den Suchraum, sondern versucht die Differenzen zwischen Lösungen von Algorithmen zu minimieren bzw. zu maximieren. Zu beachten ist dabei, dass ein solcher Ansatz erheblich aufwendiger ist. Jede zu evaluierende Probleminstanz muss dann nicht nur von einem, sondern von zwei Algorithmen gelöst werden. Das verdoppelt die Anzahl der durchzuführenden Simulationen pro paarweisem Regressionsmodell. Die Erstellung der Datenbasis für ein Regressionsmodell, genutzt von $S_{reg}(f(x))$, erfordert 24 Millionen Simulationen. Eine ähnlich umfangreiche Datenbasis für ein paarweises Regressionmodell für $S_{p-reg}(f(x))$ erfordert demzufolge 48 Millionen Simulationen. Bei der Verwendung von paarweisen Modellen werden zudem mehr Modelle für eine Entscheidung benötigt (vgl. Unterabschnitt 6.1.1), was den Aufwand bei der Generierung einer Trainingsdatenbasis zusätzlich erhöht. Aufgrund des Erfolgs des Modells $S_{reg}(f(x))$, kann der zusätzliche Aufwand für die Generierung einer Datenbasis für paarweise Regressionsmodelle vermieden werden.

In Unterabschnitt 6.1.2 zeigt sich, dass das Selektionsmodell $S_{reg}(f(x))$ basierend auf allen Problemeigenschaften weniger erfolgreich Algorithmen im Vergleich zu dem Modell selektiert, das nur die acht wichtigsten Eigenschaften verwendet. Die Vorhersagegenauigkeiten der verschiedenen Regressionsmodelle sind dabei sehr ähnlich (vgl. Tabelle 6.3). Die Modelle $S_{reg}(f(x))$ sind von der Qualität der verwendeten Regressionsmodellen stark abhängig. Überschätzt oder unterschätzt auch nur eines der Modelle die Lösungsqualität eines Algorithmus systematisch, hat das große Auswirkungen auf das Selektionsergebnis. So zeigt sich in Tabelle 6.3, dass die Vorhersagegenauigkeit der Modelle schwankt. Teilweise ist der mittlere quadratische Fehler für Modelle, die alle Eigenschaften betrachten größer, teilweise niedriger als für Modelle, die nur die acht wichtigsten Eigenschaften betrachten. Aufgrund dieser Schwankungen kann das unterschiedliche Abschneiden der Selektionsmodelle $S_{reg}(f(x))$ erklärt werden. Zudem gilt, dass die wichtigsten Eigenschaften als Durchschnitt über alle Algorithmen ermittelt sind (vgl. Unterabschnitt 5.2.1). Es ist nicht auszuschließen, dass für unterschiedliche Algorithmen verschiedene Eigenschaften einen großen Einfluss auf die Vorhersage für die Lösungsqualität haben. Mit einer algorithmenabhängigen Auswahl von Problemeigenschaften für das Erstellen der Regressionmodelle kann die Vorhersagegenauigkeit der einzelnen Modelle weiter erhöht und damit das Selektionsergebnis weiter verbessert werden.

Bei der Evaluation der Modelle mit bekannten Probleminstanzen kann das erfolgreichste Selektionsmodell den besten Algorithmus um 4,15%-Punkte unterbieten. Dynamische Instanzen, die aus bekannten statischen Probleminstanzen erzeugt werden, aber nicht

am Training oder der Evaluation der Regressionsmodelle beteiligt sind, werden als bekannte Probleminstanzen bezeichnet (vgl. Unterabschnitt 6.1.3). Für unbekannte Instanzen ist das Selektionsmodell nach wie vor erfolgreich, kann aber den besten Algorithmus nur um knapp einen Prozentpunkt unterbieten (vgl. Tabelle 6.8). Für eine Auswahl von unbekannten Instanzen kann noch eine Verbesserung um 2,2%-Punkte erreicht werden. Es zeigt sich aber, dass die Vorhersagegenauigkeit der Regressionsmodelle für unbekannte Instanzen schlechter ist, als für bekannte (vgl. Tabelle 6.7). Für diesen Umstand kann es verschiedene Gründe geben, die im Folgenden diskutiert werden. Grundsätzlich stellt sich die Frage, inwieweit sich die gelernten Zusammenhänge zwischen Problemeigenschaften und Lösungsqualität auf der Grundlage der in Unterabschnitt 5.1.2 beschriebenen Datenbasis verallgemeinern lassen. Die statischen Probleminstanzen repräsentieren gängige Strukturen wie gleichverteilte oder geclusterte Kundenanfragen (vgl. Unterabschnitt 5.1.2). Zusätzlich umfassen die bekannten statischen Instanzen auch solche, bei denen die Anfragen in künstlichen Mustern angeordnet sind. Grundsätzlich zeigt sich, dass durch die getroffene Auswahl von Probleminstanzen, Regressionsmodelle trainiert werden können, die über die verwendeten Instanzen hinaus, Algorithmen erfolgreich selektieren können. Abschließend kann aber keine Aussage darüber getroffen werden, ob andere bekannte Instanzen eine bessere Datenbasis für die Regressionsmodelle bilden. Gleiches gilt für die im Experiment berücksichtigten Werte für den DOD (vgl. Unterabschnitt 5.1.2). Es kann keine Aussage darüber getroffen werden, ob das Berücksichtigen weiterer oder anderer Werte eine Verbesserung der Selektion von Algorithmen bewirkt. Wird die Fähigkeit zur Verallgemeinerung der Regressionsmodelle diskutiert, muss die Frage aufgeworfen werden, wie repräsentativ die berechnete Lösungsqualität für einen Algorithmus für eine Probleminstanz ist. Für die Evaluation einer Probleminstanz werden für jeden Algorithmus beide Startrichtung betrachtet. Für jede Richtung werden 20 zufällige Zuordnungen von der dynamischen Anfrage auf den Zeitpunkt des Erscheinens der Anfrage berücksichtigt. Die Probleminstanz CMT12 modelliert 100 Kundenanfragen (vgl. Unterabschnitt 5.1.2). Bei der Generierung von dynamische Probleminstanzen mit einem DOD von 0,2 werden 20 Anfragen dynamisch. Diese 20 Kundenanfragen können in 2,4 Trillionen verschiedenen Zuordnungen (20!) angeordnet werden. Von der Menge der möglichen Zuordnungen wird also nur ein Bruchteil betrachtet. Es kann nicht ausgeschlossen werden, dass die 20 zufällig ausgewählten Zuordnungen nicht repräsentativ für den Algorithmus für die Probleminstanz sind. Werden die Regressionsmodelle mithilfe von nicht repräsentativen Lösungsqualitäten für Instanzen erzeugt, kann die Fähigkeit der Modelle zur Verallgemeinerung eingeschränkt sein. Zusätzliche Simulationen mit unterschiedlichen Zuordnungen können das Vertrauen in die Lösungsqualität für eine Probleminstanz erhöhen.

Kapitel 7

Fazit und Ausblick

Das letzte Kapitel widmet sich dem Fazit und dem Ausblick der gesamten Arbeit. In Abbildung 7.1, die den umgesetzten Lösungsansatz der Arbeit visualisiert, liegt aus diesem Grund der Fokus auf allen Komponenten.

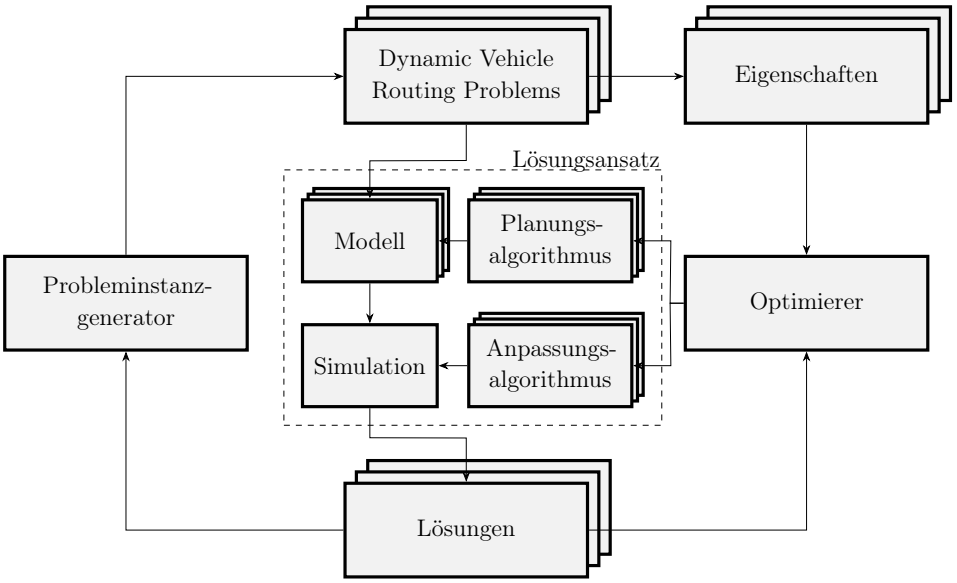

Abbildung 7.1: Lösungsansatz mit Fokus auf allen Komponenten

Abschnitt 7.1 greift die in Abschnitt 1.2 formulierten Zielstellungen für diese Arbeit auf und erläutert die Umsetzung der Ziele. Zusätzlich wird der bei der Realisierung geleistete wissenschaftliche Beitrag diskutiert. In Abschnitt 7.2 wird nochmals der Bezug der Arbeit zu dynamischen Informationen einer Problemstellung hergestellt und erläutert. Abschließend diskutiert Abschnitt 7.3 mögliche Weiterentwicklungen und wirft zusätzliche wissenschaftliche Fragestellungen im weiteren Kontext der vorliegenden Arbeit auf.

7.1 Zusammenfassung und wissenschaftlicher Beitrag

Um die Bedürfnisse unserer modernen Gesellschaft zu befriedigen, müssen mehr und mehr dynamische Routingprobleme gelöst werden. Aus diesem Grund widmet sich die vorliegende Arbeit dieser Problemstellung. Aufgrund der wachsenden Anzahl von Lösungsansätzen wird die Frage nach dem geeignetsten Algorithmus für eine konkrete Problemstellung aufgeworfen. Die Fragestellung ist durch die Aussagen der No-Free-Lunch-Theoreme legitimiert. Diese zeigen, dass die durchschnittliche Lösungsqualität aller Heuristiken für alle Probleminstanzen identisch ist. Für die Erarbeitung der Fragestellung wird der in Abbildung 1.5 dargestellte Ansatz, der sich am Framework für das *Algorithm Selection Problem* (ASP) orientiert, erarbeitet. Das Hauptziel der vorliegenden Arbeit ist die Umsetzung des erarbeiteten Ansatzes. Für die strukturierte Realisierung dieses Ziels werden Unterziele formuliert, die im Folgenden zusammengefasst werden. Zusätzlich wird der im Zuge dieser Dissertation erarbeitete wissenschaftliche Beitrag erläutert.

1. **Evaluation vorhandener Modelle und Simulatoren, Implementierung eines Simulationsmetamodells und eines Simulators**

Das Kapitel 2 erläutert in Unterabschnitt 2.1.7 neben Benchmarkproblemstellungen auch vorhandene Metamodelle, die die Abbildung von Routingproblemen unterstützten. In Mayer et al. (2016) widmet sich der Autor dieser Arbeit diesen Modellen ausführlich. Es wird gezeigt, dass die vorhandenen Metamodelle die dynamische Variante des VRP nicht abbilden können. Aus diesem Grund wird im Zuge dieser Arbeit das VRP-REP hin zum Dynamic-VRP-REP-Modell erweitert (vgl. Abschnitt 3.3). Es ist unter der Apache-Lizenz 2.0 auf Github veröffentlicht (Mayer, 2018). Aus den Ergebnissen der umfangreichen Literaturrecherche ergibt sich außerdem, dass veröffentlichte Lösungsansätze für das DVRP fast ausschließlich auf eigene Implementierungen von Simulationen zurückgreifen. Diese sind meist wenig erläutert, sehr problemspezifisch konzipiert und stehen der wissenschaftlichen Gemeinschaft nur eingeschränkt zur Verfügung (vgl. Kapitel 3). Im Zuge dieser Arbeit wird aus diesem Grund ein VRP-Simulator (vgl. Abschnitt 3.1) mit Simulationsmetamodell (vgl. Abschnitt 3.2) entwickelt. Der Simulator und das Metamodell sind ebenfalls auf Github unter der Apache-Lizenz 2.0 veröffentlicht (Mayer, 2018). Eine erste Version des Simulators wurde dem wissenschaftlichen Publikum in Mayer et al. (2016) präsentiert. Das Metamodell unterstützt eine Vielzahl von verschiedenen Variationen des VRP und ist nach einer aktuellen Taxonomie von Lahyani et al. (2015) klassifiziert. Das Metamodell und der Simulator ermöglichen der wissenschaftlichen Gemeinschaft eine schnelle und effiziente Realisierung und Erprobung eigener Lösungsansätze für das DVRP ohne den Mehraufwand der Implementierung einer Simulationsumgebung.

2. Bewertung bekannter Eigenschaften und Einführung neuer Problemeigenschaften

Problemeigenschaften, die die Dynamik in DVRP-Instanzen beschreiben, werden in der Literatur nur sehr eingeschränkt diskutiert, obwohl die Dynamik einen großen Einfluss auf die Lösungsqualität von Algorithmen hat. In Unterabschnitt 2.1.3 wird gezeigt, dass sich eine Vielzahl von unterschiedlichen Probleminstanzen allein mithilfe der bekannten Problemeigenschaften nicht unterscheiden lassen. Für die Umsetzung der Hauptzielstellung ist die Unterscheidung von Instanzen aber unerlässlich. Nur mithilfe von Problemeigenschaften lassen sich Instanzen voneinander differenzieren und anhand von Gemeinsamkeiten oder Unterschieden gruppieren. Das ist die Voraussetzung für die Ableitung von Wissen über den Zusammenhang der Beschaffenheit einer Problemstellung und der Lösungsqualität eines Algorithmus. Aus diesem Grund werden in Abschnitt 4.1 181 Problemeigenschaften, die die Dynamik einer DVRP-Instanz charakterisieren und auf bekannten Eigenschaften für das TSP basieren (vgl. Unterabschnitt 2.2.2), eingeführt. Die Eigenschaften werden unabhängig von Algorithmenselektion evaluiert und Probleminstanzen mit minimalen bzw. maximalen Ausprägungen werden visualisiert. Die Evaluation zeigt die Eignung für die Unterscheidung von verschiedene Typen von Probleminstanzen. Anhand der Visualisierungen werden Muster von statischen und dynamischen Anfragen abgeleitet, die mit den Ausprägungen der Eigenschaften einhergehen. Auf der Grundlage der erarbeiteten Muster können auch Ausprägungen in unbekannten Problemstellungen abgeschätzt werden. Das erlaubt eine manuelle Selektion von Algorithmen auch ohne Werkzeugunterstützung. Die in dieser Arbeit eingeführten Eigenschaften erlauben erstmals die Unterscheidung von Probleminstanzen anhand ihrer Dynamik. Das ermöglicht einen fairen Vergleich und die zielgerichtete Konstruktion von Algorithmen. Die Vergleichbarkeit von Lösungsansätzen wird in Abschnitt 4.3 diskutiert. Zusätzlich wird mithilfe der eingeführten Eigenschaften das Bewerten und Erzeugen von Testinstanzen für Lösungsansätze unterstützt. Für die Identifikation robuster und universell einsetzbarer Algorithmen müssen Instanzen zur Verfügung stehen, die verschiedenste Eigenschaften aufweisen. Alle in dieser Arbeit berechneten Problemeigenschaften basieren auf den Koordinaten der Kundenanfragen. Eine Verallgemeinerung, mit der alle numerischen Merkmale einer Anfrage für die Berechnung einer Eigenschaft herangezogen werden können, wird in Abschnitt 4.1.4 kurz diskutiert.

3. Konstruktion und Umsetzung eines Algorithmus für Instanzen mit bestimmten Ausprägungen von Eigenschaften

Eine Zielstellung der vorliegenden Arbeit ist die zielgerichtete Konstruktion eines Algorithmus, der für Probleminstanzen mit bestimmten Ausprägungen von Eigenschaften erfolgreicher ist, im Vergleich zu konkurrierenden Algorithmen. In Unterabschnitt 4.2.7 werden verschiedene Ansätze von Planungsalgorithmen untersucht. Es wird hergeleitet, dass das Bedienen von dynamischen Kundenanfragen am Rand der Probleminstanz in

einer nahe optimalen Rundreise zu erheblichen Mehrkosten führt. Eine flexibel geplante Route erweist sich hier als robuster. Unterabschnitt 4.2.7 führt den Spiralplanungsalgorithmus ein, der eine spiralförmige Rundreise für alle statischen Kundenanfragen plant. Die Spiralfahrt erweist sich als flexibel für das Einplanen für Kundenanfragen am Rand der Instanz. In Unterabschnitt 4.1.3 zeigt sich, dass dynamische Kundenanfragen am Rand der Probleminstanz auf einen hohen Wert des *Location based Degree of Dynamism* (LDOD) und der Flächen-Eigenschaft (FE) hinweisen. Aus diesem Grund wird die These 4.2.1 formuliert, dass der Spiralalgorithmus erfolgreicher als konkurrierende Algorithmen für große Ausprägungen der Eigenschaften LDOD und FE ist. Die Evaluation der Ergebnisse des Data-Farming-Experiments bestätigt die aufgestellte These. In Unterabschnitt 5.1.3 wird gezeigt, dass Probleminstanzen existieren, die die Variationen des Spiralalgorithmus am erfolgreichsten lösen (vgl. Abbildung 5.9). Abschnitt 5.2 untersucht die Beziehungen zwischen Problemeigenschaften und Lösungsansätzen. Es kann nachgewiesen werden, dass der Spiralalgorithmus besonders erfolgreich Probleminstanzen mit hohem LDOD und FE löst (vgl. Abbildung 5.17). Durch den Erfolg des Spiralalgorithmus wird zusätzlich gezeigt, dass Problemeigenschaften und das Wissen über den Zusammenhang zwischen Lösungsqualität und Beschaffenheit von Instanzen gezielt für die Konstruktion von Algorithmen genutzt werden können.

4. **Konzeption und Implementierung eines Ansatzes zum zielgerichteten Konstruieren von Probleminstanzen**

Das Wissen über die Zusammenhänge zwischen Lösungsqualität und Beschaffenheit einer Probleminstanz muss von gesammelten Erfahrungen abgeleitet werden. Für den Aufbau einer Erfahrungs- bzw. Datenbasis wird in Abschnitt 5.1 ein umfangreiches Data-Farming-Experiment vorgestellt. Das Experiment wird mithilfe des vorgestellten Probleminstanzgenerators realisiert (vgl. Unterabschnitt 5.1.1). Für eine aussagekräftige Evaluation der Probleminstanzen durch einen Algorithmus müssen Besonderheiten von dynamischen Problemstellungen berücksichtigt werden. Unterabschnitt 5.1.2 erläutert diese und stellt Konzepte für die Realisierung vor. Im Zuge des Data-Farming-Experiments werden 420.000 Probleminstanzen mit über 456 Millionen Simulationen evaluiert. Die generierte Erfahrungsbasis, aus der Wissen extrahiert werden kann, umfasst >50 GB Daten.

5. **Untersuchung der Beziehungen und Zusammenhänge zwischen Problemeigenschaften und Lösungsqualität von Algorithmen und damit Erarbeitung der Grundlagen für die manuelle Selektion von Algorithmen**

Bei der automatisierten Extraktion von Wissen aus Erfahrungen mit Methoden des maschinellen Lernens verbergen sich die gewonnen Erkenntnisse häufig in den verwendeten Modellen. Es ist, zum Beispiel, schwer zu rekonstruieren, welche Parameter einen Einfluss auf das Ergebnis von künstlichen neuronalen Netzen haben. Um

das Wissen über die Beziehungen zwischen der Beschaffenheit einer Problemstellung und der Lösungsqualität von Algorithmen über diese Arbeit und die implementierten Modelle hinaus nutzbar zu machen, werden in Abschnitt 5.2 die Ergebnisse des Data-Farming-Experiments mit Methoden der Data-Science untersucht. Es werden Problemeigenschaften identifiziert, die einen hohen Einfluss auf die Lösungsqualität von Algorithmen haben. Zusätzlich werden Wertebereiche der Eigenschaften aufgezeigt, für die Algorithmen besonders erfolgreich sind. Mithilfe der Ergebnisse der Analysen aus Unterabschnitt 5.2.2 kann erstmalig auch eine rudimentäre manuelle Selektion von Algorithmen für das DVRP durchgeführt werden. So kann ein Planer auch ohne Werkzeugunterstützung günstige Algorithmen für reale Problemstellungen auswählen.

6. **Umsetzung und Evaluation eines Optimierers, der auf der Grundlage von gelernten Zusammenhängen zwischen Lösungsqualität und Problemeigenschaften automatisiert Entscheidungen für Algorithmen mit hoher Lösungsqualität bei gegebenen Problemeigenschaften schlussfolgert**

In Kapitel 6 werden zwei Modelle für die automatisierte Selektion von Algorithmen für das DVRP vorgestellt. Es wird gezeigt, dass das Selektionsmodell basierend auf Regressionmodellen, die die Lösungsqualität von Algorithmen für Instanzen vorhersagen, zuverlässig den Algorithmus für Probleminstanzen auswählt, der am erfolgreichsten ist. Damit wird erstmalig gezeigt, dass die automatisierte Selektion von Algorithmen auch für Online-Probleme am Beispiel des DVRP erfolgreich ist. In Experimenten werden neben den Möglichkeiten auch die Grenzen der Selektionsmodelle erörtert (vgl. Unterabschnitt 6.1.3). Es zeigt sich, dass Wissen aus der generierten Datenbasis auch für gänzlich unbekannte Problemstellungen und Dynamiken abgeleitet werden kann. Werden unbekannte Probleme mit unbekannten Dynamiken kombiniert versagen die Selektionsmodelle. Die Erfolge und Beschränkungen werden in einer umfangreichen Diskussion in Abschnitt 6.2 erläutert. Dabei werden Ergebnisse der Kapitel 5 und 6 aufgearbeitet.

Zusammenfassend zeigt sich, dass mit der erfolgreichen Realisierung aller Unterziele, also der Umsetzung des in Abbildung 1.5 eingeführten Ansatzes, die Fragestellung nach dem passenden Algorithmus für eine konkrete Probleminstanz auch für das DVRP beantwortet werden kann. Bei der Bearbeitung der Ziele sind Artefakte und Konzepte entstanden, die einen wesentlichen wissenschaftlichen Beitrag in der Domäne Fahrzeugwegeplanung leisten.

7.2 Dynamische Informationen

In der vorliegenden Arbeit werden unter anderem wichtige Problemeigenschaften für die Auswahl von Algorithmen für das DVRP identifiziert. In die Berechnung dieser

Eigenschaften fließen Informationen über die dynamischen Ereignisse. In einer realen Problemstellung sind diese Informationen a priori nicht verfügbar. Die Anwendung von Algorithmenselektion erfordert demzufolge Prognosen über das Verhalten der dynamischen Komponenten. Hier unterstützt die vorliegende Arbeit, indem erstmalig Grundlagen erarbeitet werden, die einen Hinweis darauf geben, für welche Eigenschaften Prognosemodelle erstellt werden müssen. Die Erstellung der Modelle kann, zum Beispiel, auf historischen Daten oder Expertenwissen basieren. Die in Abschnitt 4.1 erarbeiteten Visualisierungen für die Problemeigenschaften können Experten dabei unterstützten. Die in dieser Arbeit identifizierten Eigenschaften konzentrieren sich auf den Ort einer dynamischen Kundenfrage. Die vorliegende Arbeit zeigt, dass, zum Beispiel, über das zeitliche Verhalten der Anfragen keine Prognosemodelle vorhanden sein müssen, um erfolgreich Algorithmen auszuwählen.

Die erfolgreiche Auswahl von Algorithmen für Online-Optimierungsprobleme ist wesentlich von der Güte der Prognosen über die identifizierten Problemeigenschaften abhängig. Stehen keine Modelle zur Verfügung oder ist die Qualität dieser unbekannt oder schlecht, muss auf eine Selektion von Algorithmen verzichtet werden. In einem solchen Fall empfiehlt sich die Anwendung eines robusten Algorithmus. Ein solcher Algorithmus löst diverse Probleminstanzen gut, wird aber häufig nicht die besten Ergebnisse erzielen. Für die Identifikation eines solchen Algorithmus müssen ebenfalls die in dieser Arbeit eingeführten Problemeigenschaften herangezogen werden. Die Eigenschaften ermöglichen die gezielte Generierung von signifikant unterschiedlichen Probleminstanzen, die als Testinstanzen für Algorithmen verwendet werden können. Zufällig oder willkürlich erzeugte Instanzen, wie sie aktuell häufig verwendet werden, stellen keine solide Testgrundlage für die Identifikation eines geeigneten Algorithmus dar.

7.3 Ausblick

In Abschnitt 6.2 werden bereits eine Reihe von möglichen Weiterentwicklungen der vorliegenden Arbeit diskutiert. So ist es grundsätzlich vorstellbar, dass, zum Beispiel, weitere Selektionsmodelle auf der Grundlage der bestehenden Datenbasis evaluiert werden. Ebenfalls wird die Realisierung der Datenbasis in Abhängigkeit der verschiedenen Modelle erläutert. Auch bietet das Design des Data-Farming-Experiments weitere Entwicklungsmöglichkeiten. Hier kann, zum Beispiel, untersucht werden, ob das Lösen von noch mehr bzw. anderen Probleminstanzen einen positiven Effekt auf die Selektion von Algorithmen hat. Für die industrielle Anwendung des vorgestellten Ansatzes wäre es zudem förderlich weitere Algorithmen in das Selektionsmodell aufzunehmen. Der Aufbau eines quelloffenen Algorithmenportfolios, das von verschiedenen Wissenschaftlern gepflegt wird, wäre in diesem Zusammenhang wünschenswert. Bei der Bearbeitung der in Abschnitt 7.1 zusammengefassten Zielstellungen dieser Arbeit werden eine Reihe weiterer Frage- und Problemstellungen für zukünftige Forschungsvorhaben identifiziert. Diese werden im Folgenden kurz erläutert.

Der verwendete Planungsalgorithmus ist für die Bearbeitung eines DVRP elementar (vgl. Abschnitt 5.2). Der Ansatz der vorliegenden Arbeit ist es, den Planungsalgorithmus in Kombination mit dem Anpassungsalgorithmus für eine Problemstellung auf der Grundlage von Problemeigenschaften zu selektieren. Die wichtigsten wissenschaftlichen Beiträge der vorliegenden Arbeit sind die Identifikation wichtiger Eigenschaften, auf der eine solche Selektion erfolgreich stattfindet, und die Analyse der Beziehungen zwischen Algorithmen und Beschaffenheit von Instanzen. Der selektierte Lösungsansatz wird für die Bearbeitung der gesamten Problemstellung verwendet. Diese einmalig getroffene Entscheidung wird beim Ausrollen der Problemlösung nicht angepasst, grundsätzlich wäre das aber vorstellbar. Mithilfe einer totalen Planrevision (vgl. Unterabschnitt 2.1.5), ausgelöst durch einen Anpassungsalgorithmus, kann die initiale Planung komplett revidiert werden. Aus, zum Beispiel, einer initial geplanten flexiblen Rundreise kann jederzeit eine optimale Reise konstruiert werden. Die Frage, warum geplante initiale Rundreisen angepasst werden sollten, lässt sich mithilfe der Prognosemodelle begründen. Während der Problemdurchführung können die Modelle hinsichtlich ihrer Qualität validiert werden. Weichen Vorhersagen von der Realität ab können die Prognosemodelle angepasst werden. Auf der Grundlage der neuen Erkenntnisse kann erneut ein passender Planungs- aber auch Anpassungsalgorithmus selektiert werden. Denkbar wäre es auch ausschließlich den Anpassungsalgorithmus auf der Grundlage von Problemeigenschaften in Abhängigkeit einer dynamischen Anfrage auszuwählen. Bei der Realisierung eines solchen erweiterten Lösungsansatzes müssen die folgenden Fragestellungen in einer zukünftigen wissenschaftlichen Arbeit adressiert werden.

- Können die vorhandenen Problemeigenschaften für das DVRP auch während des Ausrollens der Problemlösung sinnvoll gemessen werden?
- Wie aussagekräftig sind diese Eigenschaften für unvollständige Probleminstanzen?
- Gibt es weitere Problemeigenschaften für das DVRP, die sich aus dem Ausrollen der Problemlösung ergeben?
- Haben diese Eigenschaften einen Einfluss auf die Qualität von Algorithmen?
- Wie können Prognosemodelle durch das Auftreten von dynamischen Kundenanfragen korrigiert bzw. aktualisiert werden?
- Wie erfolgreich sind korrigierte initiale Rundreisen, die zu einem Teil flexibel und zum anderen Teil optimal geplant sind?

Die vorliegende Arbeit konzentriert sich auf die grundlegendste Variante des DVRP, das DTSP (vgl. Abschnitt 2.3). Alle entwickelten Problemeigenschaften werden basierend auf den Koordinaten der Kundenanfragen berechnet. So sind alle erarbeiteten Eigenschaften für alle Variationen des DVRP berechenbar. Grundsätzlich kann in zukünftigen wissenschaftlichen Arbeiten untersucht werden, ob die analysierten Beziehungen zwischen Eigenschaften und Problemschwere für weitere Varianten des DVRP ebenfalls

gültig sind. Es ist durchaus anzunehmen, das sich nicht nur die Lage von Anfragen auf die Schwere von Problemstellungen für Algorithmen auswirkt, sondern, zum Beispiel, auch die Kapazität, die durch eine Anfrage definiert wird (vgl. Unterabschnitt 2.1.3). Die vorliegende Arbeit formuliert in Unterabschnitt 4.1.4 eine allgemeine Problemeigenschaft, die auf der Grundlage beliebiger numerischer Merkmale von Kundenanfragen berechnet werden kann (vgl. Gleichung 4.6). Diese allgemeine Eigenschaft kann als Ausgangspunkt für Untersuchung weiterer Variationen des DVRP dienen. Diese zukünftigen Untersuchungen sollten die folgenden wissenschaftlichen Fragestellungen adressieren.

- Haben die in dieser Arbeit eingeführten Problemeigenschaften, die auf der Grundlage der Koordinaten der Anfragen berechnet werden, auch einen Einfluss auf die Schwere von Problemstellung anderer Variationen des DVRP?
- Eignet sich die in dieser Arbeit eingeführte verallgemeinerte Problemeigenschaft für die Selektion von Algorithmen für Probleminstanzen anderer Variationen des DVRP?

Abkürzungsverzeichnis

ACO *Ant Colony Optimization*

AGV *Automated Guided Vehicles*

AS *Ant System*

ASP *Algorithm Selection Problem*

ACO *Ant Colony Optimization*

BG Benchmarkgenerator

CPP *Chinese Postman Problem*

CTREE *Conditional Inference Trees*

DARP *Dial-A-Ride Problem*

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*

DepotLDOD *Depot LDOD*

DOD *Degree of Dynamism*

DTSP *Dynamic TSP*

DVRP *Dynamic Vehicle Routing Problem*

EDOD *Effective Degree of Dynamism*

EDOD-TW *Effective Degree of Dynamism with Time Windows*

EA Evolutionärer Algorithmus

FDC *Fitness Distance Correlation*

FE Flächen-Eigenschaft

FIFO *First In First Out*

GA Genetischer Algorithmus

GB Gigabyte

GENI *Generalized Insertion Procedure*

GPS *Global Positioning System*

GVRP *General VRP*

HVRP *Heterogeneous fleet VRP*

IRP *Inventory Routing Problem*

LA *Lookahead Algorithm*

LDOD *Location based Degree of Dynamism*

LCP *Livestock Collection Problem*

LIFO *Last In First Out*

LoVRP *Loading VRP*

LRP *Location Routing Problem*

MDVRP *Multi-Depot Vehicle Routing Problem*

MMAS *Max Min AS*

MST Minimaler Spannbaum

MTDP *Multi Terminal Delivery Problem*

OVRP *Open VRP*

PDP *Pick-up and Delivery Problem*

PDTRP *Partially Dynamic Traveling Repairman Problem*

PFA *Policy Function Approximation*

PVRDCP *Practical Vehicle Routing and Driver Scheduling Problem*

PVRP *Periodic VRP*

RO Re-Optimierung

RVRP *Rich VRP*

RVRPSim *Rich Vehicle Routing Problem Simulator*

SAT *Satisfiability Problem*

SDRAP *Stochastic Dynamic Resource Allocation Problem*

SEO *Single Event Optimization*

SMI *Supplier Managed Inventory*

SVM *Support Vector Machines*

SVRP *Stochastic VRP*

TSP *Travelling Salesman Problem*

TTRP *Truck and Trailer Routing Problem*

US *Unstringing/Stringing*

VFA *Value Function Approximation*

VRP *Vehicle Routing Problem*

VRPB *VRP with Backhauls*

VRPCTW *VRP with Coupled Time Windows*

VRPPD *VRP with Pickups and Deliveries*

SDVRP *VRP with Split Deliveries*

VRPOD *VRP with Occasional Drivers*

VRPTW *VRP with Time Windows*

VRPTWTD *VRPTW and Temporal Dependencies*

VMI *Vendor Managed Inventory*

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016), Tensorflow: A system for large-scale machine learning, *in* '12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)', pp. 265–283.

Alvarez, J. M. und Salzmann, M. (2016), Learning the number of neurons in deep networks, *in* 'Advances in Neural Information Processing Systems', pp. 2270–2278.

Archetti, C., Savelsbergh, M. und Speranza, M. G. (2016), 'The vehicle routing problem with occasional drivers', *European Journal of Operational Research* **254**(2), 472–480.

Archetti, C. und Speranza, M. G. (2012), 'Vehicle routing problems with split deliveries', *International transactions in operational research* **19**(1-2), 3–22.

Arndt, H. (2004), *Supply Chain Management*, Springer.

Attanasio, A., Cordeau, J.-F., Ghiani, G. und Laporte, G. (2004), 'Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem', *Parallel Computing* **30**(3), 377–387.

Azi, N., Gendreau, M. und Potvin, J.-Y. (2012), 'A dynamic vehicle routing problem with multiple delivery routes', *Annals of Operations Research* **199**(1), 103–112.

Baldacci, R., Battarra, M. und Vigo, D. (2008), Routing a heterogeneous fleet of vehicles, *in* 'The vehicle routing problem: latest advances and new challenges', Springer, pp. 3–27.

Beaudry, A., Laporte, G., Melo, T. und Nickel, S. (2010), 'Dynamic transportation of patients in hospitals', *OR spectrum* **32**(1), 77–107.

Bektaş, T., Repoussis, P. P. und Tarantilis, C. D. (2014), Chapter 11: Dynamic vehicle routing problems, *in* 'Vehicle Routing: Problems, Methods, and Applications, Second Edition', SIAM, pp. 299–347.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G. und Prutzman, P. J. (1983), 'Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer', *Interfaces* **13**(6), 4–23.

Beller, S. (2016), *Empirisch forschen lernen: Konzepte, Methoden, Fallbeispiele, Tipps*, Vol. 3, Hofgrefe Verlag.

Beltrami, E. J. und Bodin, L. D. (1974), 'Networks and vehicle routing for municipal waste collection', *Networks* **4**(1), 65–94.

Bent, R. und Van Hentenryck, P. (2004), Online stochastic and robust optimization, *in* 'Advances in Computer Science-ASIAN 2004. Higher-Level Decision Making', Springer, pp. 286–300.

Bent, R. und Van Hentenryck, P. (2007), Waiting and relocation strategies in online stochastic vehicle routing., *in* 'IJCAI', pp. 1816–1821.

Bentley, J. J. (1992), 'Fast algorithms for geometric traveling salesman problems', *ORSA Journal on computing* **4**(4), 387–411.

Berbeglia, G., Cordeau, J.-F. und Laporte, G. (2012), 'A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem', *INFORMS Journal on Computing* **24**(3), 343–355.

Berhan, E., Beshah, B., Kitaw, D. und Abraham, A. (2014), 'Stochastic vehicle routing problem: A literature survey', *Journal of Information & Knowledge Management* **13**(03), 1450022.

Bertsimas, D. J. und Simchi-Levi, D. (1996), 'A new generation of vehicle routing research: robust algorithms, addressing uncertainty', *Operations Research* **44**(2), 286–304.

Bertsimas, D. J. und Van Ryzin, G. (1991), 'A stochastic and dynamic vehicle routing problem in the euclidean plane', *Operations Research* **39**(4), 601–615.

Bianchi, L. (2000), 'Notes on dynamic vehicle routing-the state of the art'.

Bierwirth, C., Mattfeld, D. C. und Watson, J.-P. (2004), Landscape regularity and random walks for the job-shop scheduling problem, *in* 'European Conference on Evolutionary Computation in Combinatorial Optimization', Springer, pp. 21–30.

Borodin, A. und El-Yaniv, R. (2005), *Online computation and competitive analysis*, cambridge university press.

Börsenblatt (2019), 'Amazon meldet rund 17 milliarden euro umsatz in deutschland'. **URL:** *www.boersenblatt.net/2019-02-06-artikel-bilanz_fuer_2018.1594276.html/*

Branke, J., Middendorf, M., Noeth, G. und Dessouky, M. (2005), 'Waiting strategies for dynamic vehicle routing', *Transportation science* **39**(3), 298–312.

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**(1), 5–32.

Bridle, J. S. (1990), Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, *in* 'Neurocomputing', Springer, pp. 227–236.

Brotcorne, L., Laporte, G. und Semet, F. (2003), 'Ambulance location and relocation models', *European journal of operational research* **147**(3), 451–463.

Bruns, A. (1998), Zweistufige Standortplanung unter Bercksichtigung von Tourenplanungsaspekten - Primale Heuristiken und Lokale Suchverfahren, PhD thesis, Sankt Gallen University.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. und Schulenburg, S. (2003), Hyper-heuristics: An emerging direction in modern search technology, *in* 'Handbook of metaheuristics', Springer, pp. 457–474.

Campbell, A., Clarke, L., Kleywegt, A. und Savelsbergh, M. (1998), The inventory routing problem, *in* 'Fleet management and logistics', Springer, pp. 95–113.

Campbell, A. M. und Wilson, J. H. (2014), 'Forty years of periodic vehicle routing', *Networks* **63**(1), 2–15.

Censor, Y. (1977), 'Pareto optimality in multiobjective problems', *Applied Mathematics and Optimization* **4**(1), 41–59.

Cheeseman, P. C., Kanefsky, B. und Taylor, W. M. (1991), Where the really hard problems are., *in* 'IJCAI', Vol. 91, pp. 331–340.

Cheung, B. K.-S., Choy, K., Li, C.-L., Shi, W. und Tang, J. (2008), 'Dynamic routing model and solution methods for fleet management with mobile technologies', *International Journal of Production Economics* **113**(2), 694–705.

Chollet, F. et al. (2015), 'Keras', `https://keras.io`.

Christiansen, C. H. und Lysgaard, J. (2007), 'A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands', *Operations Research Letters* **35**(6), 773–781.

Christofides, N. (1976), 'The vehicle routing problem', *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle* **10**(V1), 55–70.

Christofides, N. und Beasley, J. E. (1984), 'The period routing problem', *Networks* **14**(2), 237–256.

Colorni, A., Dorigo, M., Maniezzo, V. et al. (1991), Distributed optimization by ant colonies, *in* 'Proceedings of the first European conference on artificial life', Vol. 142, Paris, France, pp. 134–142.

Cordeau, J.-F., Gendreau, M. und Laporte, G. (1997), 'A tabu search heuristic for periodic and multi-depot vehicle routing problems', *Networks* **30**(2), 105–119.

Cornillier, F., Laporte, G., Boctor, F. F. und Renaud, J. (2009), 'The petrol station replenishment problem with time windows', *Computers & Operations Research* **36**(3), 919–935.

Crevier, B., Cordeau, J.-F. und Laporte, G. (2007), 'The multi-depot vehicle routing problem with inter-depot routes', *European Journal of Operational Research* **176**(2), 756–773.

Croes, G. A. (1958), 'A method for solving traveling-salesman problems', *Operations research* **6**(6), 791–812.

Dan, B., Zhu, W., Li, H., Sang, Y. und Liu, Y. (2013), 'Dynamic optimization model and algorithm design for emergency materials dispatch', *Mathematical Problems in Engineering* **2013**.

Dantzig, G. B. und Ramser, J. H. (1959), 'The truck dispatching problem', *Management science* **6**(1), 80–91.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F. und Vogel, U. (2011), 'Vehicle routing with compartments: applications, modelling and heuristics', *OR spectrum* **33**(4), 885–914.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M. und Francois, S. (2000), *The VRP with pickup and delivery*, Montréal: Groupe d'études et de recherche en analyse des décisions.

Dohn, A., Rasmussen, M. S. und Larsen, J. (2011), 'The vehicle routing problem with time windows and temporal dependencies', *Networks* **58**(4), 273–289.

Dorigo, M. (1992), Ant Colony Optimization for vehicle routing problem, PhD thesis, PhD thesis, Politecnico di Milano, Milan, Italy.

Dorigo, M. und Stützle, T. (2003), The ant colony optimization metaheuristic: Algorithms, applications, and advances, *in* 'Handbook of metaheuristics', Springer, pp. 250–285.

Drexl, M. (2012), 'Synchronization in vehicle routing - a survey of vrps with multiple synchronization constraints', *Transportation Science* **46**(3), 297–316.

Dror, M. und Trudeau, P. (1989), 'Savings by split delivery routing', *Transportation Science* **23**(2), 141–145.

Esser, K. und Kurte, J. (2018), *KEP-Studie 20018 - Analyse des Marktes in Deutschland*, Vol. 14, Bundesverband Paket und Expresslogistik e. V.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise., *in* 'Kdd', Vol. 96, pp. 226–231.

Fabri, A. und Recht, P. (2006), 'On dynamic pickup and delivery vehicle routing with several time windows and waiting times', *Transportation Research Part B: Methodological* **40**(4), 335–350.

Ferrucci, F., Bock, S. und Gendreau, M. (2013), 'A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods', *European Journal of Operational Research* **225**(1), 130–141.

Fleischmann, B. (1990), The vehicle routing problem with multiple use of vehicles, Technical report, Fachbereich Wirtschaftswissenschaften, Universitdt Hamburg.

Fleischmann, B., Gnutzmann, S. und Sandvoß, E. (2004), 'Dynamic vehicle routing based on online traffic information', *Transportation science* **38**(4), 420–433.

Flood, M. M. (1956), 'The traveling-salesman problem', *Operations Research* **4**(1), 61–75.

Fränti, P. und Sieranoja, S. (2018), 'K-means properties on six clustering benchmark datasets'.
URL: *http://cs.uef.fi/sipu/datasets/*

Fügenschuh, A. (2006), 'The vehicle routing problem with coupled time windows', *Central European Journal of Operations Research* **14**(2), 157–176.

Gan, Z., Tao, L. und Ying, Q. (2013), 'Automated guide vehicles dynamic scheduling based on annealing genetic algorithm', *Indonesian Journal of Electrical Engineering and Computer Science* **11**(5), 2508–2515.

Garrido, P. und Riff, M. C. (2010), 'Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic', *Journal of Heuristics* **16**(6), 795–834.

Gendreau, M., Guertin, F., Potvin, J.-Y. und Séguin, R. (2006), 'Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries', *Transportation Research Part C: Emerging Technologies* **14**(3), 157–174.

Gendreau, M., Guertin, F., Potvin, J.-Y. und Taillard, E. (1999), 'Parallel tabu search for real-time vehicle routing and dispatching', *Transportation science* **33**(4), 381–390.

Gendreau, M., Hertz, A. und Laporte, G. (1992), 'New insertion and postoptimization procedures for the traveling salesman problem', *Operations Research* **40**(6), 1086–1094.

Gendreau, M., Laporte, G. und Semet, F. (2001), 'A dynamic model and parallel tabu search heuristic for real-time ambulance relocation', *Parallel computing* **27**(12), 1641–1653.

Gendreau, M. und Tarantilis, C. D. (2010), *Solving large-scale vehicle routing problems with time windows: The state-of-the-art*, Cirrelt Montreal.

Ghiani, G., Laporte, G., Manni, E. und Musmanno, R. (2008), 'Waiting strategies for the dynamic and stochastic traveling salesman problem', *International Journal of Operations Research* **5**(4), 233–241.

Glover, F. (1986), 'Future paths for integer programming and links to artificial intelligence', *Computers and operations research* **13**(5), 533–549.

Godfrey, G. A. und Powell, W. B. (2002), 'An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times', *Transportation Science* **36**(1), 21–39.

Goel, A. und Gruhn, V. (2008), 'A general vehicle routing problem', *European Journal of Operational Research* **191**(3), 650–660.

Goel, A. und Vidal, T. (2013), 'Hours of service regulations in road freight transport: An optimization-based international assessment', *Transportation science* **48**(3), 391–412.

Goetschalckx, M. und Jacobs-Blecha, C. (1989), 'The vehicle routing problem with backhauls', *European Journal of Operational Research* **42**(1), 39–51.

Golden, B. L., Wasil, E. A., Kelly, J. P. und Chao, I.-M. (1998), The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results, *in* 'Fleet management and logistics', Springer, pp. 33–56.

Grötschel, M., Krumke, S. O., Rambau, J., Winter, T. und Zimmermann, U. T. (2001), Combinatorial online optimization in real time, *in* 'Online optimization of large scale systems', Springer, pp. 679–704.

Grötschel, M., Lovász, L. und Schrijver, A. (2012), *Geometric algorithms and combinatorial optimization*, Vol. 2, Springer Science & Business Media.

Grötschel, M. und Padberg, M. W. (1977), 'Lineare charakterisierungen von travelling salesman problemen', *Zeitschrift für Operations Research* **21**(1), 33–64.

Grötschel, M. und Yuan, Y.-x. (2012), 'Euler, mei-ko kwan, königsberg, and a chinese postman', *Optimization Stories* p. 43.

Guyon, I. und Elisseeff, A. (2003), 'An introduction to variable and feature selection', *Journal of machine learning research* **3**(Mar), 1157–1182.

Hecht-Nielsen, R. (1992), Theory of the backpropagation neural network, *in* 'Neural networks for perception', Elsevier, pp. 65–93.

Hentenryck, P. V. und Bent, R. (2009), *Online stochastic combinatorial optimization*, The MIT Press.

Hevner, A. R., March, S. T., Park, J. und Ram, S. (2004), 'Design science in information systems research', *MIS quarterly* **28**(1), 75–105.

Hofmann-Wellenhof, B., Lichtenegger, H. und Collins, J. (2012), *Global positioning system: theory and practice*, Springer Science & Business Media.

Hothorn, T., Hornik, K. und Zeileis, A. (2006), 'Unbiased recursive partitioning: A conditional inference framework', *Journal of Computational and Graphical statistics* **15**(3), 651–674.

Hvattum, L. M., Løkketangen, A. und Laporte, G. (2006), 'Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic', *Transportation Science* **40**(4), 421–438.

Ichoua, S., Gendreau, M. und Potvin, J.-Y. (2000), 'Diversion issues in real-time vehicle dispatching', *Transportation Science* **34**(4), 426–438.

Iori, M. (2005), Metaheuristic algorithms for combinatorial optimization problems, PhD thesis, Springer.

Iori, M. und Martello, S. (2010), 'Routing problems with loading constraints', *Top* **18**(1), 4–27.

Jones, T. und Forrest, S. (1995), Fitness distance correlation as a measure of problem didculty for genetic algorithms, *in* 'Proceedings of the 6th international conference on genetic algorithms', Vol. 95, pp. 184–192.

Jozefowiez, N., Semet, F. und Talbi, E.-G. (2008), 'Multi-objective vehicle routing problems', *European journal of operational research* **189**(2), 293–309.

Kadioglu, S., Malitsky, Y., Sellmann, M. und Tierney, K. (2010), Isac-instance-specific algorithm configuration., *in* 'ECAI', Vol. 215, pp. 751–756.

Kallehauge, B., Larsen, J., Madsen, O. B. und Solomon, M. M. (2005), Vehicle routing problem with time windows, *in* 'Column generation', Springer, pp. 67–98.

Kilby, P., Prosser, P. und Shaw, P. (1998), 'Dynamic vrps: A study of scenarios', *University of Strathclyde, Technical Report* pp. 1–11.

Kingma, D. P. und Ba, J. L. (2014), Adam: A method for stochastic optimization, *in* 'Proc. 3rd Int. Conf. Learn. Representations'.

Kirby, D. (1959), 'Is your fleet the right size?', *Journal of the Operational Research Society* **10**(4), 252–252.
URL: *https://doi.org/10.1057/jors.1959.25*

Knight, K. und Hofer, J. (1968), 'Vehicle scheduling with timed and connected calls: A case study', *Journal of the Operational Research Society* **19**(3), 299–310.

Koch, J. (1997), Die analyse des kombinierten verkehrs, *in* 'Die Entwicklung des Kombinierten Verkehrs', Springer, pp. 73–195.

Kraus, S. und Giselbrecht, C. (2015), 'Shareconomy: Das disruptive geschäftsmodell des teilens', *ZfKE–Zeitschrift für KMU und Entrepreneurship* **63**(1), 77–93.

Kriesel, D. (2007), *Ein kleiner Überblick über Neuronale Netze.*
URL: *erhältlich auf http://www.dkriesel.com*

Kruskal, J. B. (1956), 'On the shortest spanning subtree of a graph and the traveling salesman problem', *Proceedings of the American Mathematical society* **7**(1), 48–50.

Krypczyk, V. (2010), 'Nachbarschaftssucheverfahren für dynamische pickup-und delivery-probleme'.

Lahyani, R., Khemakhem, M. und Semet, F. (2015), 'Rich vehicle routing problems: From a taxonomy to a definition', *European Journal of Operational Research* **241**(1), 1–14.

Laporte, G. (2009), 'Fifty years of vehicle routing', *Transportation Science* **43**(4), 408–416.

Larsen, A. (2000), The dynamic vehicle routing problem, PhD thesis, Technical University of Denmark.

Larsen, A., Madsen, O. B. und Solomon, M. M. (2007), Classification of dynamic vehicle routing systems, *in* 'Dynamic Fleet Management', Springer, pp. 19–40.

Larsen, A., Madsen, O. B. und Solomon, M. M. (2008), Recent developments in dynamic vehicle routing systems, *in* 'The Vehicle Routing Problem: Latest Advances and New Challenges', Springer, pp. 199–218.

Law, A. M. und Kelton, W. D. (2000), *Simulation modeling and analysis*, Vol. 3, McGraw-Hill New York.

Lawler, E. L. (1985), 'The traveling salesman problem: a guided tour of combinatorial optimization', *Wiley-Interscience Series in Discrete Mathematics* .

Lenstra, J. K. und Kan, A. (1981), 'Complexity of vehicle routing and scheduling problems', *Networks* **11**(2), 221–227.

Li, J.-Q., Mirchandani, P. B. und Borenstein, D. (2009*a*), 'A lagrangian heuristic for the real-time vehicle rescheduling problem', *Transportation Research Part E: Logistics and Transportation Review* **45**(3), 419–433.

Li, J.-Q., Mirchandani, P. B. und Borenstein, D. (2009*b*), 'Real-time vehicle rerouting problems with time windows', *European Journal of Operational Research* **194**(3), 711–727.

Lin, C., Choy, K. L., Ho, G. T., Chung, S. H. und Lam, H. (2014), 'Survey of green vehicle routing problem: past and future trends', *Expert Systems with Applications* **41**(4), 1118–1138.

Lin, S. und Kernighan, B. W. (1973), 'An effective heuristic algorithm for the traveling-salesman problem', *Operations research* **21**(2), 498–516.

Lindauer, M., Hoos, H. H., Hutter, F. und Schaub, T. (2015), 'Autofolio: An automatically configured algorithm selector', *Journal of Artificial Intelligence Research* **53**, 745–778.

Lund, K., Madsen, O. B. und Rygaard, J. M. (1996), 'Vehicle routing with varying degree of dynamism.'.

MacKay, D. J. (2003), *Information theory, inference and learning algorithms*, Cambridge university press.

Macready, W. G. und Wolpert, D. H. (1996), 'What makes an optimization problem hand?', *Complexity* **1**(5), 40–46.

Maratea, M., Pulina, L. und Ricca, F. (2014), 'A multi-engine approach to answer-set programming', *Theory and Practice of Logic Programming* **14**(6), 841–868.

Mavrovouniotis, M., Müller, F. M. und Yang, S. (2015), An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem, *in* 'Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation', ACM, pp. 49–56.

Mavrovouniotis, M., Müller, F. M. und Yang, S. (2017), 'Ant colony optimization with local search for dynamic traveling salesman problems', *IEEE transactions on cybernetics* **47**(7), 1743–1756.

Mavrovouniotis, M., Yang, S. und Yao, X. (2012), A benchmark generator for dynamic permutation-encoded problems, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 508–517.

Mayer, T. (2018), 'Java based rich vehicle routing problem simulator (rvrp simulator)'. **URL:** *https://github.com/MayerTh/RVRPSimulator*

Mayer, T., Uhlig, T. und Rose, O. (2016), An open-source discrete event simulator for rich vehicle routing problems, *in* 'Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on', IEEE, pp. 1305–1310.

Mayer, T., Uhlig, T. und Rose, O. (2017), A location model for dynamic vehicle routing problems, *in* S. Wenzel und T. Peter, eds, 'Simulation in Produktion und Logistik 2017', kassel university press, Kassel, pp. 149–158.

Mayer, T., Uhlig, T. und Rose, O. (2018*a*), Simulation-based autonomous algorithm selection for dynamic vehicle routing problems with the help of supervised learning methods, *in* 'Winter Simulation Conference (WSC)', Goetheburg.

Mayer, T., Uhlig, T. und Rose, O. (2018*b*), Simulation-based evaluation of dynamic vehicle routing problem features for algorithm selection, *in* '24. Symposium Simulationstechnik Grundlagen, Methoden und Anwendungen in Modellbildung und Simulation', Hamburg.

Mei-Ko, K. (1962), 'Graphic programming using odd or even points', *Chinese Math.* **1**, 273–277.

Mendoza, J., Guéret, C., Hoskins, M., Lobit, H., Pillac, V., Vidal, T. und Vigo, D. (2014), 'VRP-REP: The vehicle routing community repository. third meeting of the euro working group on vehicle routing and logistics optimization (verolog)', *Oslo (Norway)* **56**, 77.

Mendoza, J., Guéret, C., Hoskins, M., Lobit, H., Pillac, V., Vidal, T. und Vigo, D. (2019), 'VRP-REP: The vehicle routing community repository.'.
URL: *http://www.vrp-rep.org/*

Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W. und Hamprecht, F. A. (2009), 'A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data', *BMC bioinformatics* **10**(1), 213.

Mersmann, O., Bischl, B., Bossek, J., Trautmann, H., Wagner, M. und Neumann, F. (2012), Local search and the traveling salesman problem: A feature-based characterization of problem hardness, *in* 'Learning and Intelligent Optimization', Springer, pp. 115–129.

Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J. und Neumann, F. (2013), 'A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem', *Annals of Mathematics and Artificial Intelligence* **69**(2), 151–182.
URL: *http://dx.doi.org/10.1007/s10472-013-9341-2*

Misir, M. und Sebag, M. (2013), 'Algorithm selection as a collaborative filtering problem'.

Mitrović-Minić, S., Krishnamurti, R. und Laporte, G. (2004), 'Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows', *Transportation Research Part B: Methodological* **38**(8), 669–685.

Mladenović, N. und Hansen, P. (1997), 'Variable neighborhood search', *Computers and operations research* **24**(11), 1097–1100.

Mohamad, I. B. und Usman, D. (2013), 'Standardization and its effects on k-means clustering algorithm', *Research Journal of Applied Sciences, Engineering and Technology* **6**(17), 3299–3303.

Moin, N. H. und Salhi, S. (2007), 'Inventory routing problems: a logistical overview', *Journal of the Operational Research Society* **58**(9), 1185–1194.

Mu, Q., Fu, Z., Lysgaard, J. und Eglese, R. (2011), 'Disruption management of the vehicle routing problem with vehicle breakdown', *Journal of the Operational Research Society* **62**(4), 742–749.

Munuzuri, J., Larrañeta, J. und Ibanez, J. (2005), 'Routing of delivery vehicles in a city with access time windows', *WIT Transactions on The Built Environment* **77**.

Nagy, G. und Salhi, S. (2007), 'Location-routing: Issues, models and methods', *European journal of operational research* **177**(2), 649–672.

Nair, V. und Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'Proceedings of the 27th international conference on machine learning (ICML-10)', pp. 807–814.

Nicolai, B. (2018), 'Amazon sucht paketfahrer, das auto sollen sie mitbringen'.
URL: *https://www.welt.de/wirtschaft/article185180620/Amazon-sucht-Paketfahrer-das-Auto-sollen-sie-mitbringen.html*

Novoa, C. M. (2005), Static and dynamic approaches for solving the vehicle routing problem with stochastic demands, Technical report.

Novoa, C. und Storer, R. (2009), 'An approximate dynamic programming approach for the vehicle routing problem with stochastic demands', *European Journal of Operational Research* **196**(2), 509–515.

Oppen, J. und Løkketangen, A. (2006), The livestock collection problem, *in* 'Proceedings of the 21st European conference on Operational Research. Google Scholar'.

Pankratz, G. (2002), Speditionelle transportdisposition, *in* 'Speditionelle Transportdisposition', Springer, pp. 27–59.

Peffers, K., Tuunanen, T., Rothenberger, M. A. und Chatterjee, S. (2007), 'A design science research methodology for information systems research', *Journal of management information systems* **24**(3), 45–77.

Pfahringer, B., Bensusan, H. und Giraud-Carrier, C. G. (2000), Meta-learning by landmarking various learning algorithms., *in* 'ICML', pp. 743–750.

Pillac, V., Gendreau, M., Guéret, C. und Medaglia, A. L. (2013), 'A review of dynamic vehicle routing problems', *European Journal of Operational Research* **225**(1), 1–11.

Polacek, M., Hartl, R. F., Doerner, K. und Reimann, M. (2004), 'A variable neighborhood search for the multi depot vehicle routing problem with time windows', *Journal of heuristics* **10**(6), 613–627.

Powell, W. B. (2011), *Approximate Dynamic Programming: Solving the curses of dimensionality*, Vol. 703, John Wiley & Sons.

Psaraftis, H. (1988), 'Dynamic vehicle routing problems.', *Vehicle Routing: Methods and Studies* **16**, 223–248.

Psaraftis, H. N. (1980), 'A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem', *Transportation Science* **14**(2), 130–154.

Psaraftis, H. N. (1995), 'Dynamic vehicle routing: Status and prospects', *Annals of operations research* **61**(1), 143–164.

Psaraftis, H. N., Wen, M. und Kontovas, C. A. (2016), 'Dynamic vehicle routing problems: Three decades and counting', *Networks* **67**(1), 3–31.

Pureza, V. und Laporte, G. (2008), 'Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows', *Infor* **46**(3), 165.

Raczynski, B. und Jörg, C. (2005), *Das Nokia-Communicator-Powerbuch*, Franzis.

Rand, W. M. (1971), 'Objective criteria for the evaluation of clustering methods', *Journal of the American Statistical association* **66**(336), 846–850.

Reeves, C. R. (1999), 'Landscapes, operators and heuristic search', *Annals of Operations Research* **86**, 473–490.

Regan, A. C., Mahmassani, H. S. und Jaillet, P. (1995), 'Improving efficiency of commercial vehicle operations using real-time information: potential uses and assignment strategies', *Transportation Research Record* (1493), 188–198.

Regan, A., Mahmassani, H. und Jaillet, P. (1996), 'Dynamic decision making for commercial fleet operations using real-time information', *Transportation Research Record: Journal of the Transportation Research Board* (1537), 91–97.

Rice, J. R. (1976), 'The algorithm selection problem', *Advances in computers* **15**, 65–118.

Ridge, E. und Kudenko, D. (2007), An analysis of problem difficulty for a class of optimisation heuristics, *in* 'European Conference on Evolutionary Computation in Combinatorial Optimization', Springer, pp. 198–209.

Ritzinger, U., Puchinger, J. und Hartl, R. F. (2016), 'A survey on dynamic and stochastic vehicle routing problems', *International Journal of Production Research* **54**(1), 215–231.

Russell, R., Chiang, W.-C. und Zepeda, D. (2008), 'Integrating multi-product production and distribution in newspaper logistics', *Computers & Operations Research* **35**(5), 1576–1588.

Russell, S. und Norvig, P. (2012), *Künstliche Intelligenz*, Pearson Deutschland GmbH.

Samulowitz, H. und Memisevic, R. (2007), Learning to solve qbf, *in* 'AAAI', Vol. 7, pp. 255–260.

Sanchez, S. M. (2014), Simulation experiments: better data, not just big data, *in* 'Proceedings of the 2014 Winter Simulation Conference', IEEE Press, pp. 805–816.

Schmid, V., Doerner, K. F. und Laporte, G. (2013), 'Rich routing problems arising in supply chain management', *European Journal of Operational Research* **224**(3), 435–448.

Schrage, L. (1981), 'Formulation and structure of more complex/realistic routing and scheduling problems', *Networks* **11**(2), 229–232.

Schreiber, G. A. (2004), *Telemetrie und Telematik in der Logistik:[Basistechnologien; Märkte; Produkte; Anwendungen]*, Dt. Wirtschaftsdienst.

Schröder, A. (2017), 'Die geschichte des handys - vom motorola dynatac bis google glass'.
URL: *https://www.toptarif.de/handy/wissen/handy-geschichte/*

Sim, K., Hart, E., Urquhart, N. und Pigden, T. (2015), 'An xml object model for rich vehicle routing problems'.

Sleator, D. D. und Tarjan, R. E. (1985), 'Amortized efficiency of list update and paging rules', *Communications of the ACM* **28**(2), 202–208.

Smith-Miles, K. A. (2009), 'Cross-disciplinary perspectives on meta-learning for algorithm selection', *ACM Computing Surveys (CSUR)* **41**(1), 6.

Smith-Miles, K. und Lopes, L. (2012), 'Measuring instance difficulty for combinatorial optimization problems', *Computers & Operations Research* **39**(5), 875–889.

Smith-Miles, K., van Hemert, J. und Lim, X. Y. (2010), Understanding tsp difficulty by learning from evolved instances, *in* 'International Conference on Learning and Intelligent Optimization', Springer, pp. 266–280.

Solomon, M. M. (1987), 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations research* **35**(2), 254–265.

Stadler, P. F. und Schnabl, W. (1992), 'The landscape of the traveling salesman problem', *Physics Letters A* **161**(4), 337–344.

Streeter, M., Golovin, D. und Smith, S. F. (2007), Combining multiple heuristics online, *in* 'AAAI', pp. 1197–1203.

Stutzle, T. und Hoos, H. (1997), Max-min ant system and local search for the traveling salesman problem, *in* 'Evolutionary Computation, 1997., IEEE International Conference on', IEEE, pp. 309–314.

Taillard, É. D., Laporte, G. und Gendreau, M. (1996), 'Vehicle routeing with multiple use of vehicles', *Journal of the Operational research society* **47**(8), 1065–1070.

Tibshirani, R., Walther, G. und Hastie, T. (2001), 'Estimating the number of clusters in a data set via the gap statistic', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **63**(2), 411–423.

Tillman, F. A. (1969), 'The multiple terminal delivery problem with probabilistic demands', *Transportation Science* **3**(3), 192–204.

Tilp, A. und Schiefer, M. (2017), 'Vw dieselgate. die notwendigkeit zur einfuehrung einer zivilrechtlichen sammelklage', *NZV-Neue Zeitschrift fuer Verkehrsrecht* **30**(1).

Toregas, C., Swain, R., ReVelle, C. und Bergman, L. (1971), 'The location of emergency service facilities', *Operations Research* **19**(6), 1363–1373.

Uber (2019), 'Uberpool, gemeinsam sparen'.
URL: *https://www.uber.com/de/ride/uberpool/*

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T. und Subramanian, A. (2017), 'New benchmark instances for the capacitated vehicle routing problem', *European Journal of Operational Research* **257**(3), 845–858.

Uhlig, T. (2015), *Self-Replicating Individuals*, Verlag Dr. Hut.

Ulmer, M. W. (2017), *Approximate Dynamic Programming for Dynamic Vehicle Routing*, Vol. 61, Springer.

Ulmer, M. W., Goodson, J. C., Mattfeld, D. C. und Thomas, B. W. (2017), 'Dynamic vehicle routing: Literature review and modeling framework'.

van Hemert, J. I. (2006), 'Evolving combinatorial problem instances that are difficult to solve', *Evolutionary Computation* **14**(4), 433–462.

Vilalta, R. und Drissi, Y. (2002), 'A perspective view and survey of meta-learning', *Artificial Intelligence Review* **18**(2), 77–95.

Wagner, M., Lindauer, M., Mısır, M., Nallaperuma, S. und Hutter, F. (2018), 'A case study of algorithm selection for the traveling thief problem', *Journal of Heuristics* **24**(3), 295–320.

Ward Jr, J. H. (1963), 'Hierarchical grouping to optimize an objective function', *Journal of the American statistical association* **58**(301), 236–244.

Wegener, I. (2013), *Komplexitätstheorie: Grenzen der Effizienz von Algorithmen*, Springer-Verlag.

Wen, M., Krapper, E., Larsen, J. und Stidsen, T. K. (2011), 'A multilevel variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem', *Networks* **58**(4), 311–322.

Wilson, N. H. und Colvin, N. J. (1977), *Computer control of the Rochester dial-a-ride system*, Massachusetts Institute of Technology, Center for Transportation Studies.

Wirtz, M. A. und Nachtigall, C. (2004), *Deskriptive Statistik*, Vol. 36, Beltz Juventa.

Witten, I. H., Frank, E., Hall, M. A. und Pal, C. J. (2016), *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann.

Wolpert, D. H. und Macready, W. G. (1997), 'No free lunch theorems for optimization', *IEEE transactions on evolutionary computation* **1**(1), 67–82.

Xu, L., Hutter, F., Hoos, H. H. und Leyton-Brown, K. (2008), 'Satzilla: portfolio-based algorithm selection for sat', *Journal of artificial intelligence research* **32**, 565–606.

Xu, L., Hutter, F., Hoos, H. H. und Leyton-Brown, K. (2011), Hydra-mip: Automated algorithm configuration and selection for mixed integer programming, *in* 'RCRA workshop on experimental evaluation of algorithms for solving problems with combinatorial explosion at the international joint conference on artificial intelligence (IJCAI)', pp. 16–30.

Anhang A

DVRP-Eigenschaften

Anhang A ergänzt die durchgeführte Analyse der Problemeigenschaften (vgl. Abschnitt A.2). Zusätzlich werden in Abschnitt A.1 grundlegende Berechnungsvorschriften für die in dieser Arbeit eingeführten Eigenschaften von DVRP vorgestellt.

A.1 Berechnungsvorschriften für DVRP-Eigenschaften

Nachfolgender Abschnitt schildert Berechnungsvorschriften, die für die Berechnung der in Abschnitt 4.1 eingeführten herangezogen werden.

A.1.1 Schnittwinkel

Der Schnittwinkel α zwischen dem Punkt A_1 und den Punkten A_2 und A_3 berechnet sich mithilfe der Richtungsvektoren $\vec{a_{12}}$ und $\vec{a_{13}}$ (Vergl. Abbildung A.1). Die Berechnungsvorschrift ist in Gleichung A.1 erfasst.

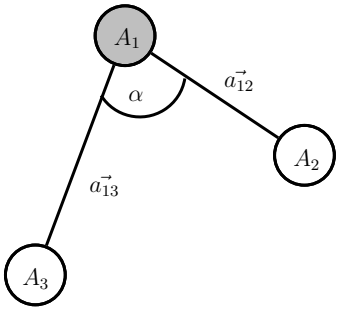

Abbildung A.1: Schnittwinkel α zwischen dem Punkt A_1 und den Punkten A_2 und A_3

$$\alpha = arccos(\frac{\vec{a_{12}} \cdot \vec{a_{13}}}{|\vec{a_{12}}| \cdot |\vec{a_{13}}|}) \tag{A.1}$$

A.1.2 Standardabweichung einer Stichprobe

Die Standardabweichung sd der Stichprobe K berechnet sich mithilfe der Messwerte x der Stichprobe, der Anzahl der Werte n und dem Erwartungswert $E(X)$ nach der in Gleichung A.2 abgebildeten Vorschrift.

$$sd(K) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - E(X))^2}{n-1}} \tag{A.2}$$

A.1.3 Variationskoeffizient

Der Variationskoeffizient $VarK$ einer Zufallsvariable X berechnet sich mithilfe der Wurzel aus der Varianz V und dem Quadrat des Erwartungswertes E. Die Berechnungsvorschrift ist in Gleichung A.3 erfasst.

$$VarK(X) = \sqrt{\frac{V(X)}{E^2(X)}} \tag{A.3}$$

A.1.4 Minimaler Spannbaum

Ein Spannbaum eines ungerichtet Graphen ist ein Teilgraph, der alle Knoten des Graphen enthält. Der Teilgraph ist dabei zusammenhängend und enthält keine geschlossenen Pfade. Ein minimaler Spannbaum von ungerichteten Graphen kann mit dem Algorithmus von Kruskal, eingeführt in Kruskal (1956), berechnet werden. Der Pseudocode des Algorithmus von Kruskal ist in Algorithm 1 aufgelistet.

Algorithm 1 Algorithmus von Kruskal

1: $E' = \varnothing$
2: L =Menge der Kanten
3: Sortieren der Kanten in L nach Kantengewicht (Kosten)
4: **while** $L \neq \varnothing$ **do**
5: e = Kante mit geringstem Kantengewicht
6: Entferne e aus L
7: **if** $Graph(V, E' \cup \{e\})$ keinen Kreis enthält **then**
8: $E' = E' \cup \{e\}$
9: **end if**
10: **end while**
11: $MST = Graph(V, E')$ ist ein minimaler Spannbaum

A.1.5 Konvexe Hülle

Die konvexe Hülle $convX$ einer Teilmenge X ist die kleinste konvexe Menge, die X enthält. Eine Menge heißt dann konvex, wenn eine Verbindungsstrecke für zwei beliebige

Punkte der Menge komplett in der Menge selbst liegt. Die konvexe Hülle kann als Schnitt aller konvexen Obermenge K von X beschrieben werden (vgl. Gleichung A.4).

$$convX = \cap K \tag{A.4}$$

A.1.6 Geometrischer Schwerpunkt

Der geometrische Schwerpunkt für komplexe Flächen kann mithilfe numerischer Integration bestimmt werden. Ein Algorithmus für die numerische Integration ist in Algorithmus 2 dargestellt. Die Methode `inArea` prüft, ob der gegebene Punkt innerhalb der Fläche liegt.

Algorithm 2 Algorithmus zur Schwerpunktberechnung mithilfe numerischer Integration

```
1: A = 0
2: mx = 0
3: my = 0
4: for x = 0 to maxX do
5:    for y = 0 to maxY do
6:       if inArea(x,y) then
7:          A = A + 1
8:          mx = mx + x
9:          my = my + y
10:      end if
11:   end for
12: end for
13: xCentroid = mx/A
14: yCentroid = my/A
```

A.2 Analyseergebnisse der DVRP-Eigenschaften

In den zwei folgenden Abschnitten sind Ergebnisse der Analyse der Wertebereiche aller Winkel- und Nächsten-Nachbar-Eigenschaften abgebildet.

Winkel-Eigenschaften

In den nachfolgenden Abbildungen A.2, A.3, A.4, A.5, A.6 und A.7 sind die Wertebereiche, der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Winkel-Eigenschaften, für die verschiedenen Typen von Probleminstanzen, vorgestellt in Abschnitt 4.1.1, abgebildet.

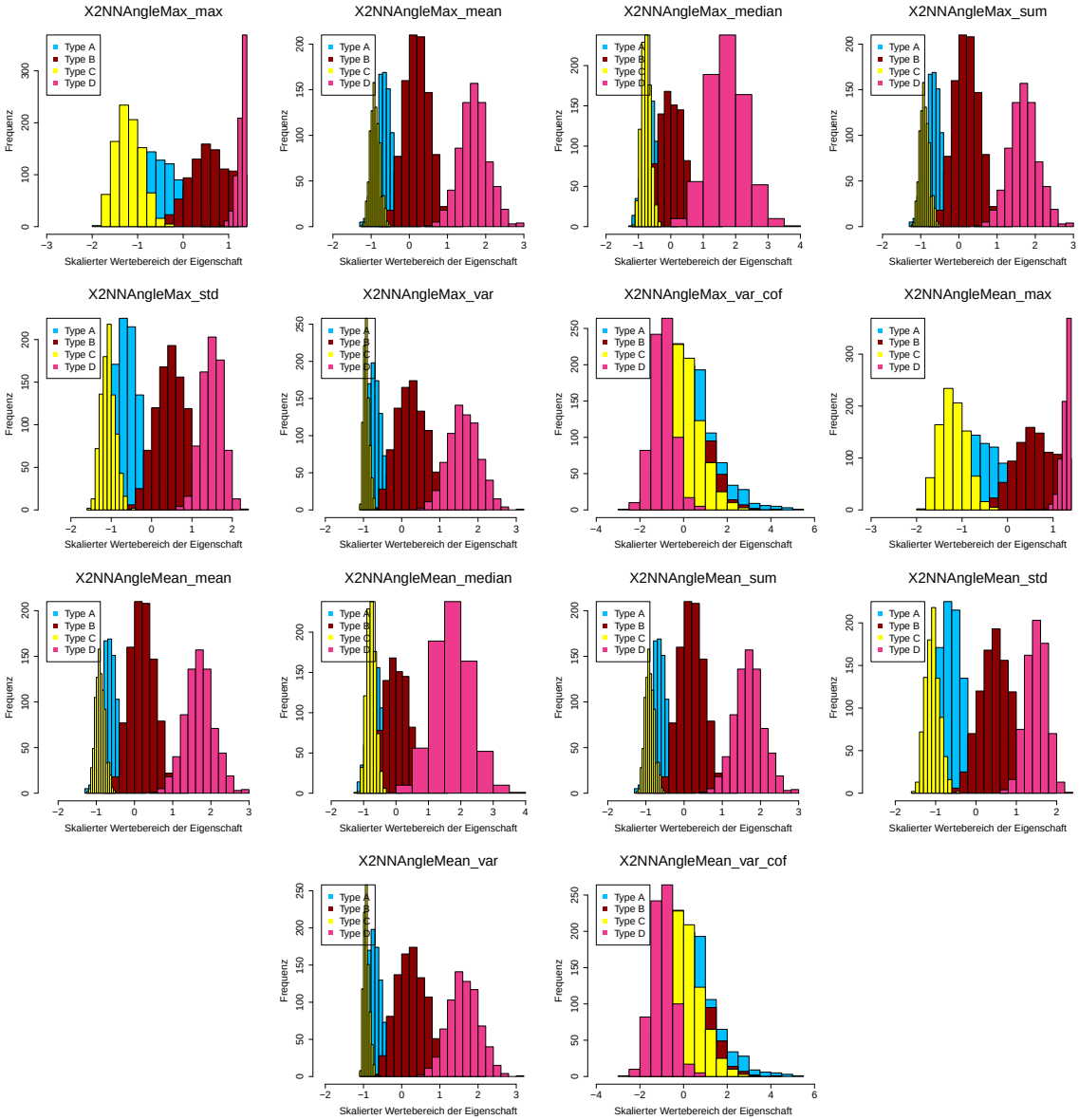

Abbildung A.2: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Winkel-Eigenschaften Max und Mean mit der Nachbarschaft N mit $n = 2$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

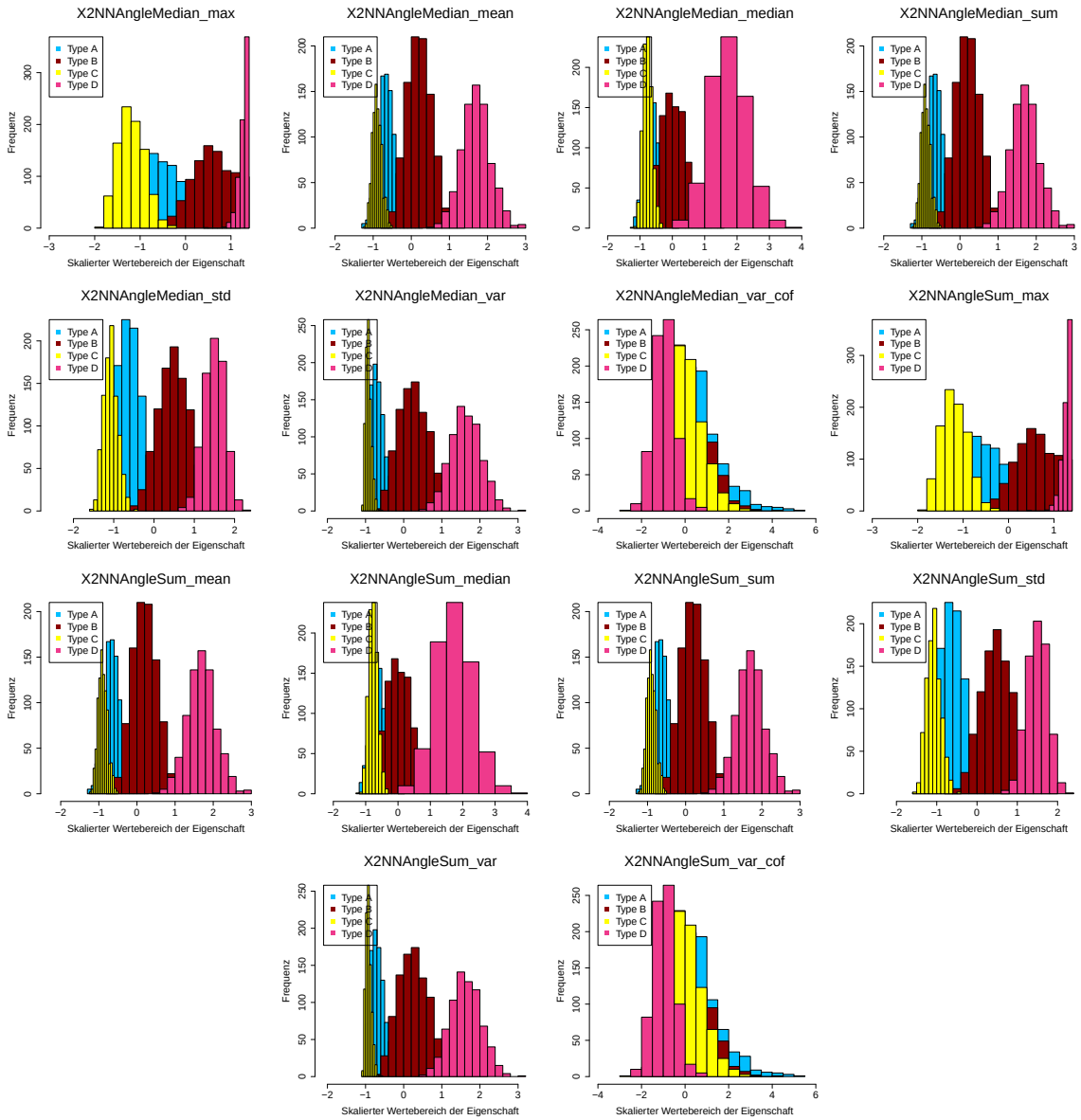

Abbildung A.3: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Winkel-Eigenschaften Median und Summe mit der Nachbarschaft N mit $n = 2$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

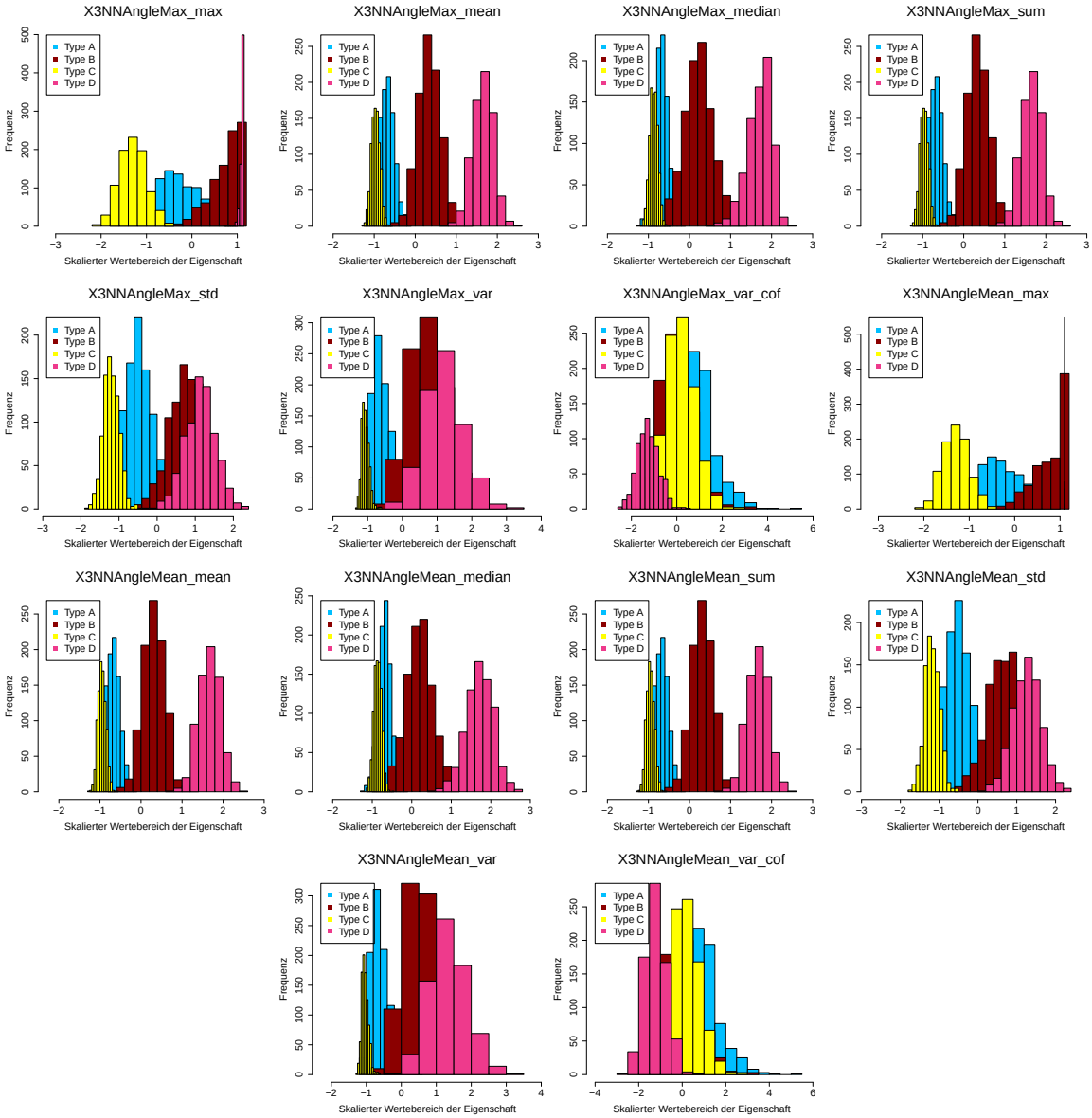

Abbildung A.4: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Winkel-Eigenschaften Max und Mean mit der Nachbarschaft N mit $n = 3$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

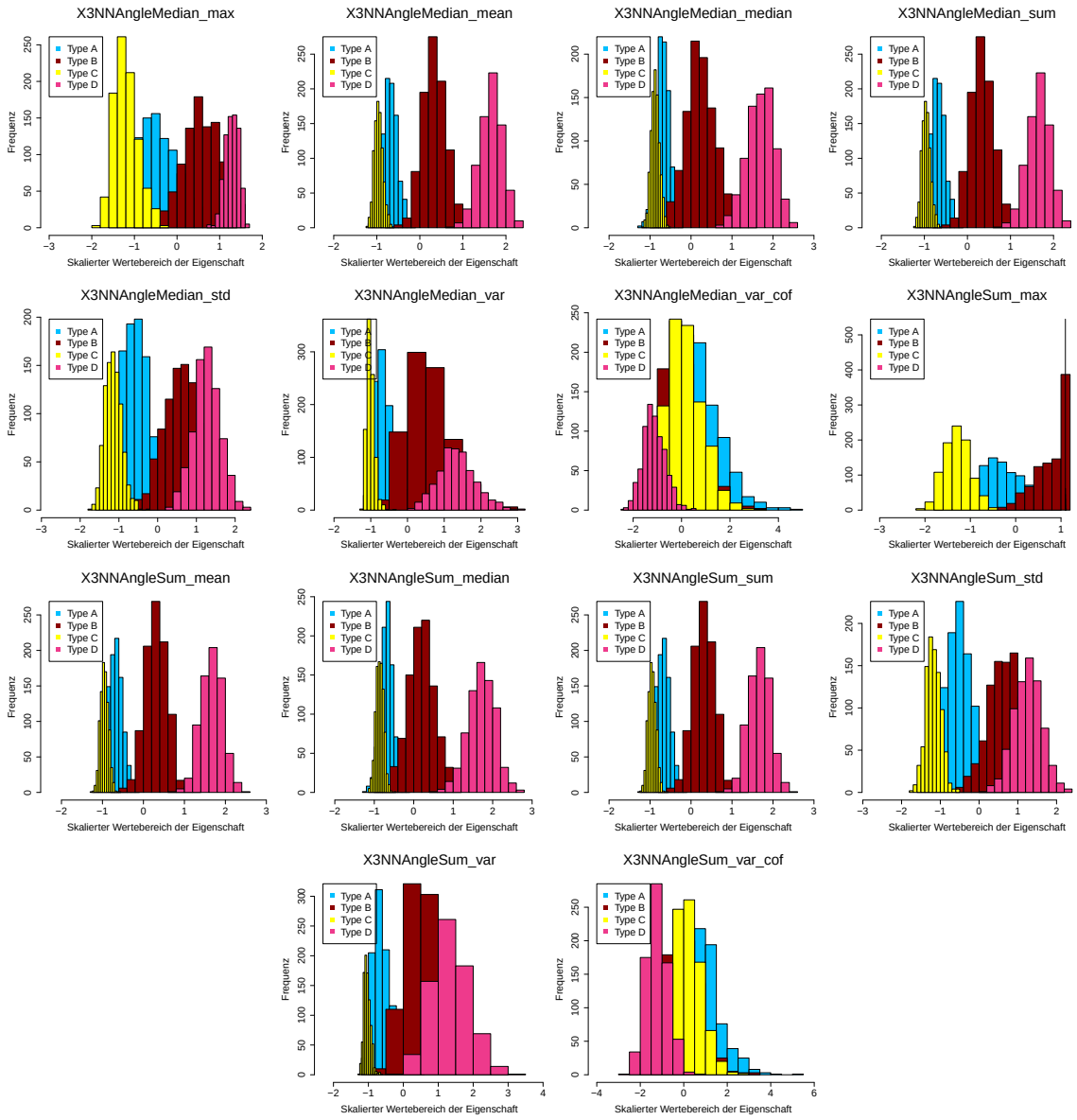

Abbildung A.5: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Winkel-Eigenschaften Median und Summe mit der Nachbarschaft N mit $n = 3$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

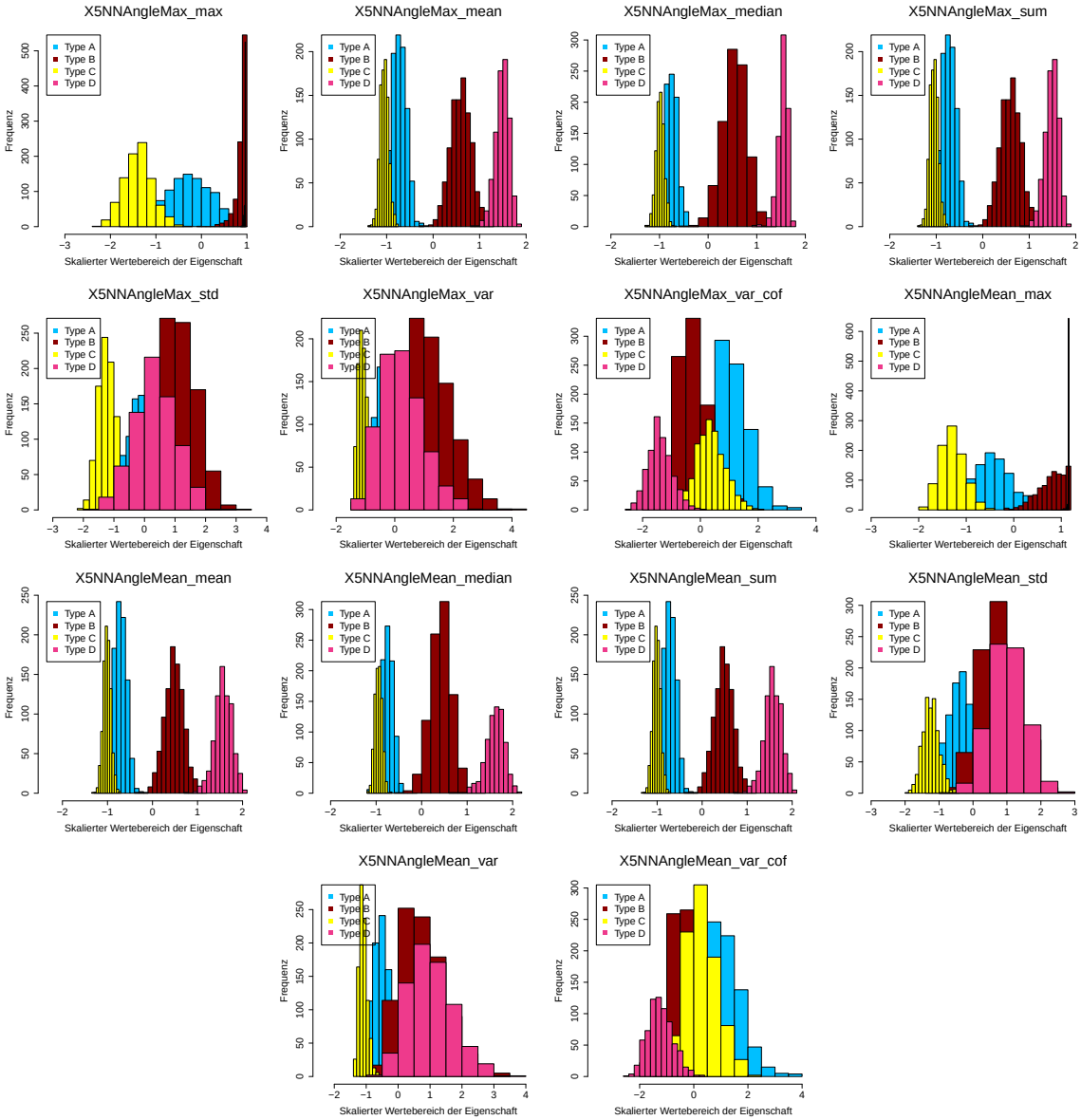

Abbildung A.6: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Winkel-Eigenschaften Max und Mean mit der Nachbarschaft N mit $n = 5$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

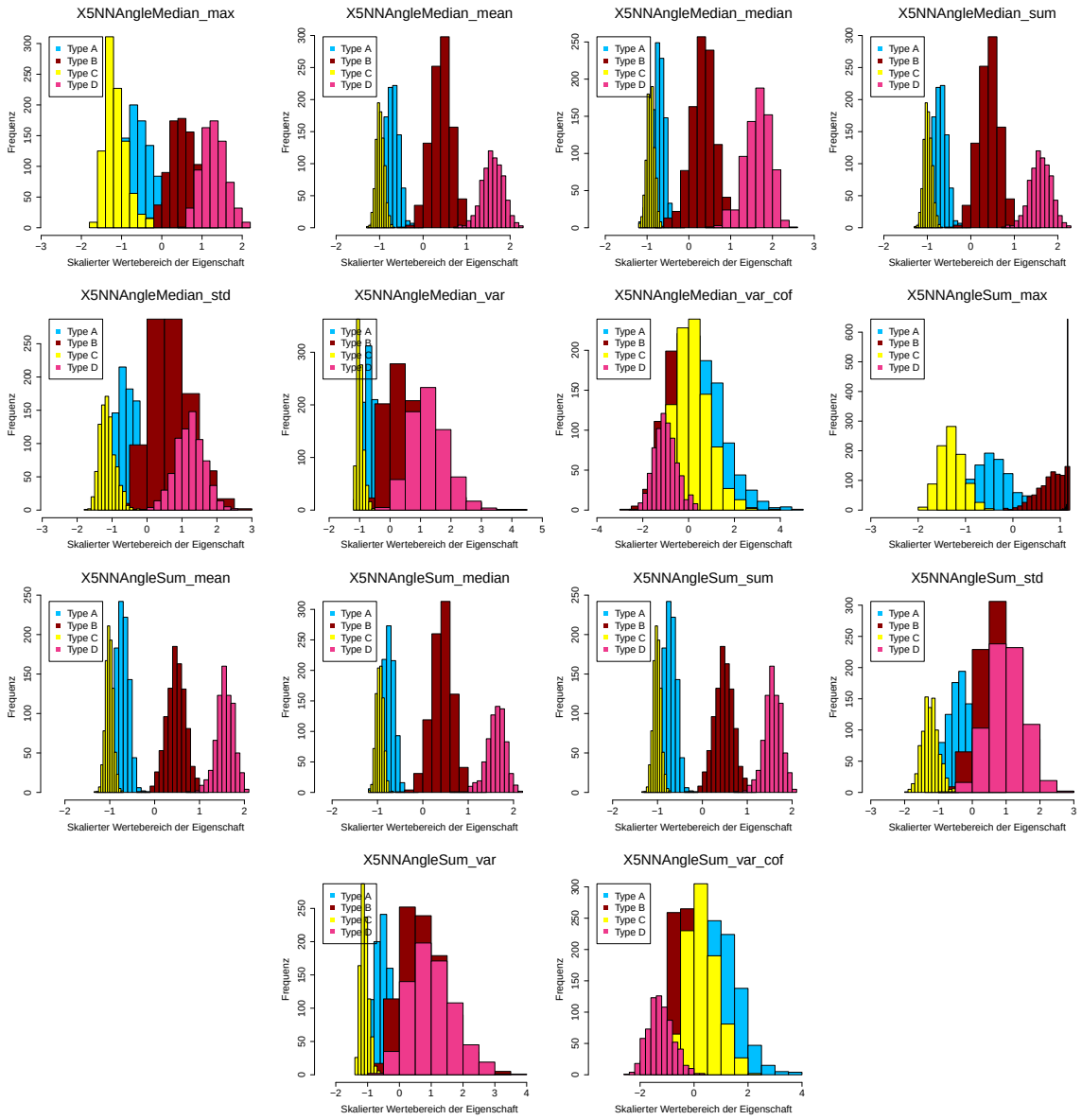

Abbildung A.7: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Winkel-Eigenschaften Median und Summe mit der Nachbarschaft N mit $n = 5$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

Nächsten-Nachbar-Eigenschaften

In den nachfolgenden Abbildungen A.8, A.9, A.10, A.11, A.12 und A.13 sind die Wertebereiche, der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Nächsten-Nachbar-Eigenschaften (NN-Eigenschaften), für die verschiedenen Typen von Probleminstanzen, vorgestellt in Abschnitt 4.1.1, abgebildet.

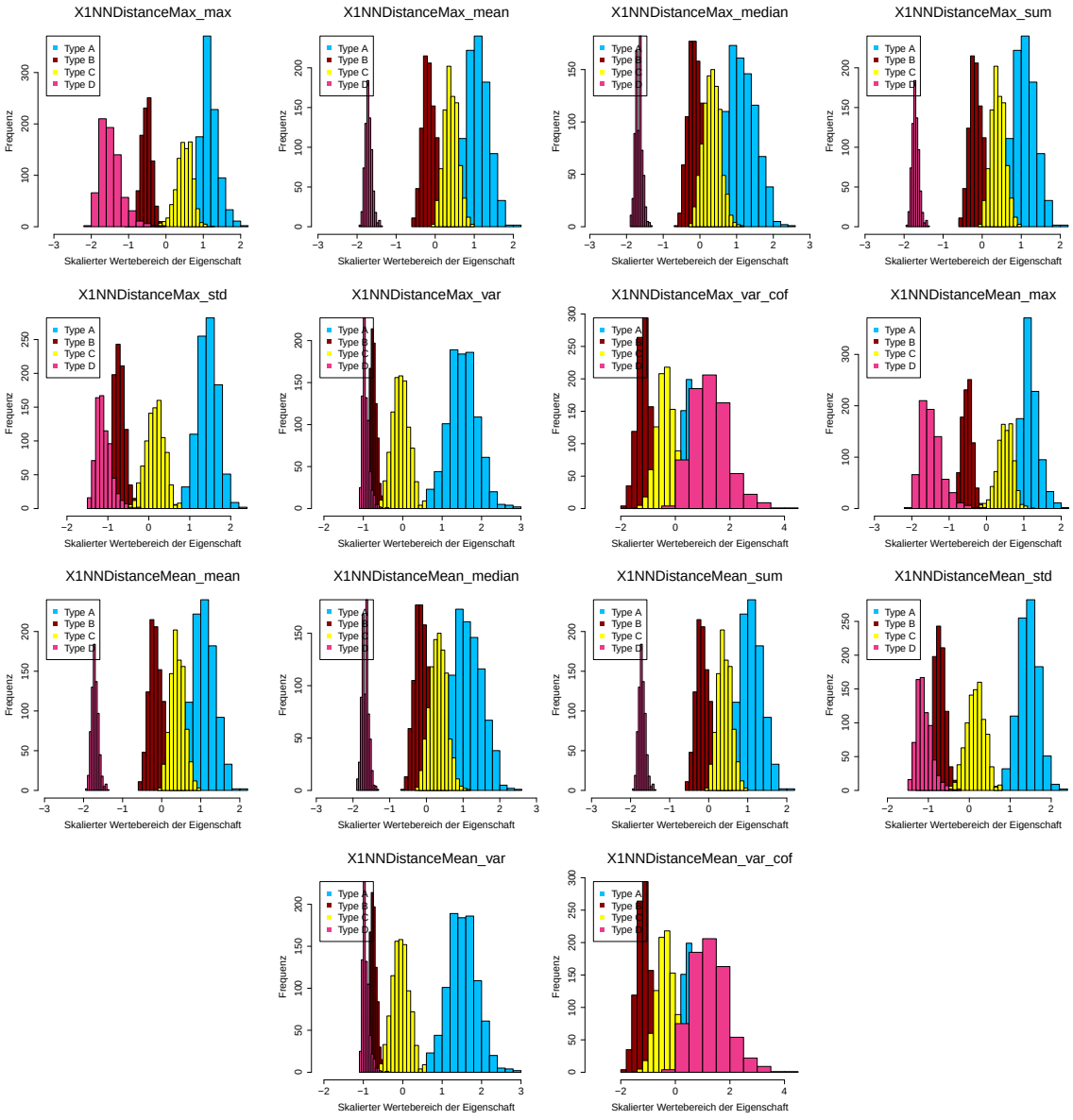

Abbildung A.8: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Nächsten-Nachbar-Eigenschaften Max und Mean mit der Nachbarschaft N mit $n = 1$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

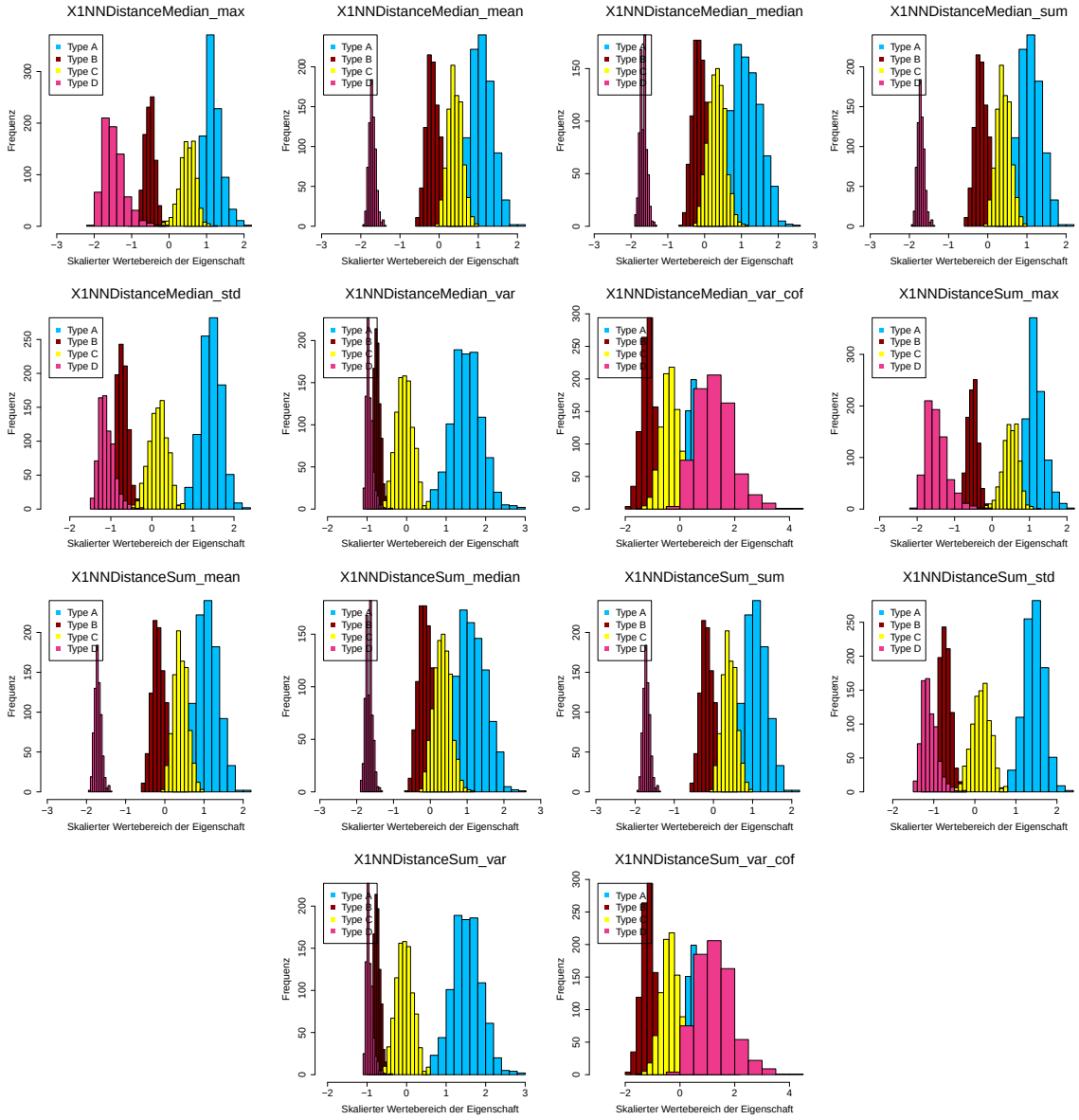

Abbildung A.9: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Nächsten-Nachbar-Eigenschaften Median und Summe mit der Nachbarschaft N mit $n = 1$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

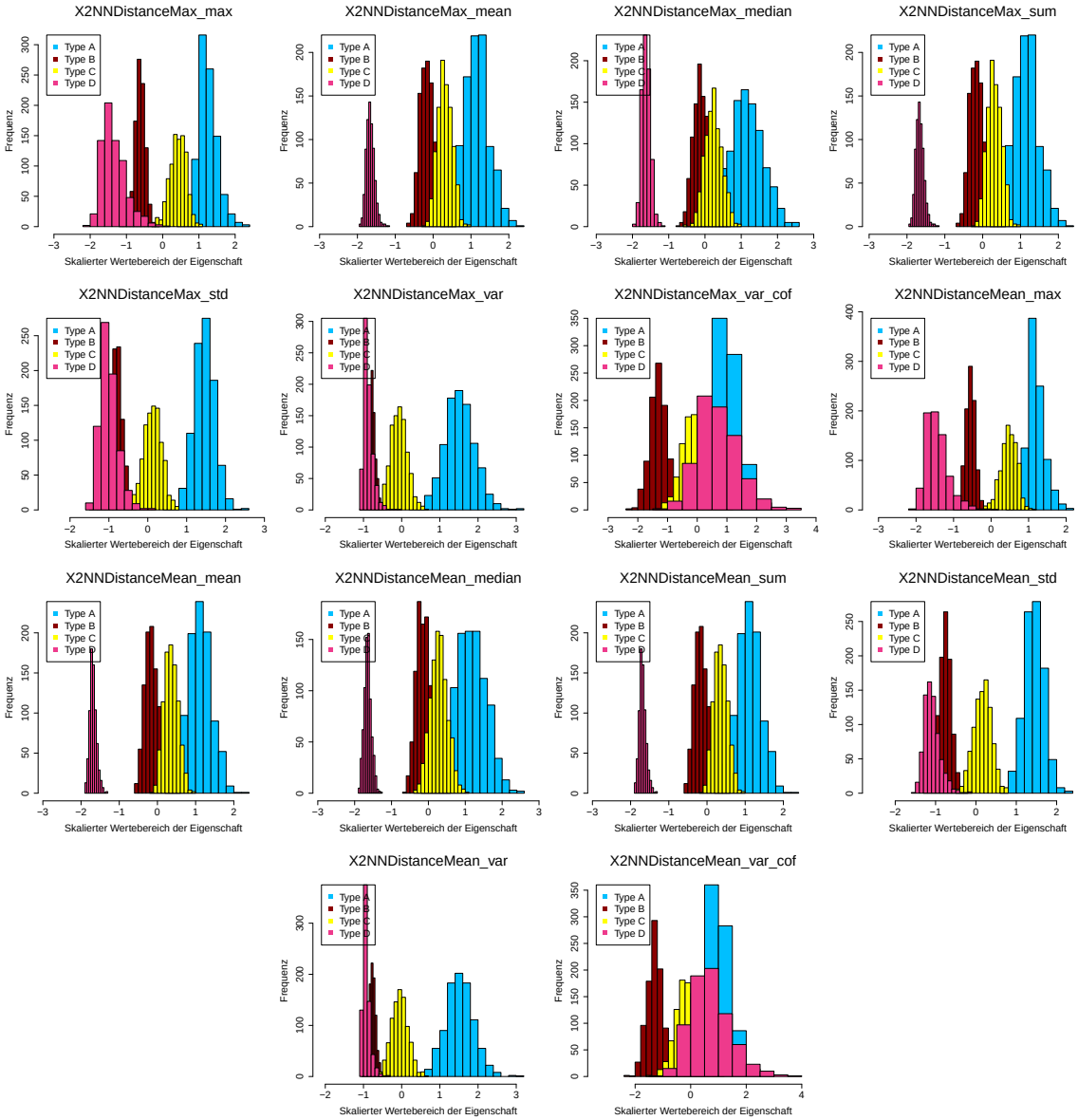

Abbildung A.10: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Nächsten-Nachbar-Eigenschaften Max und Mean mit der Nachbarschaft N mit $n = 2$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

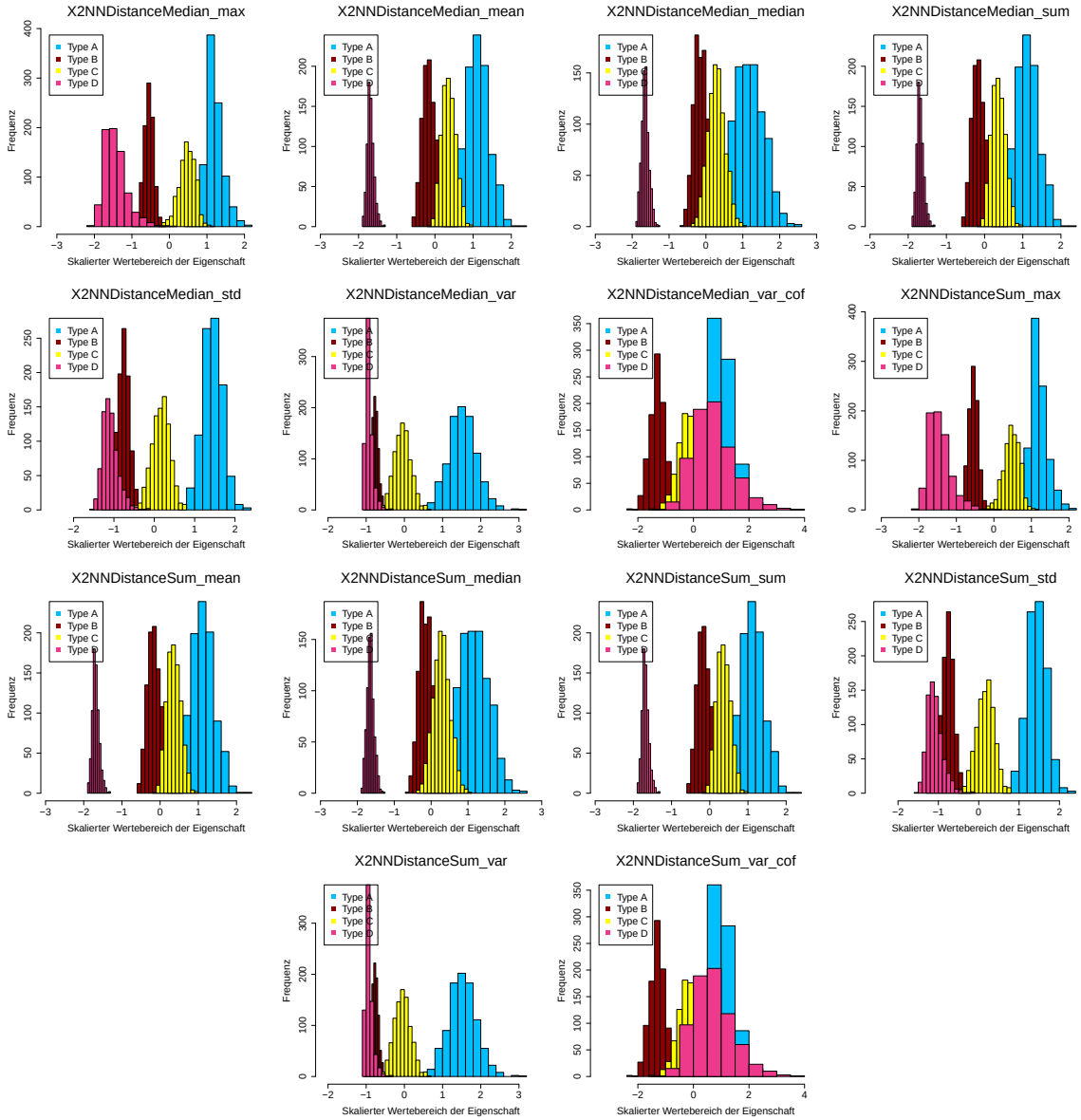

Abbildung A.11: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Nächsten-Nachbar-Eigenschaften Median und Summe mit der Nachbarschaft N mit $n = 2$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

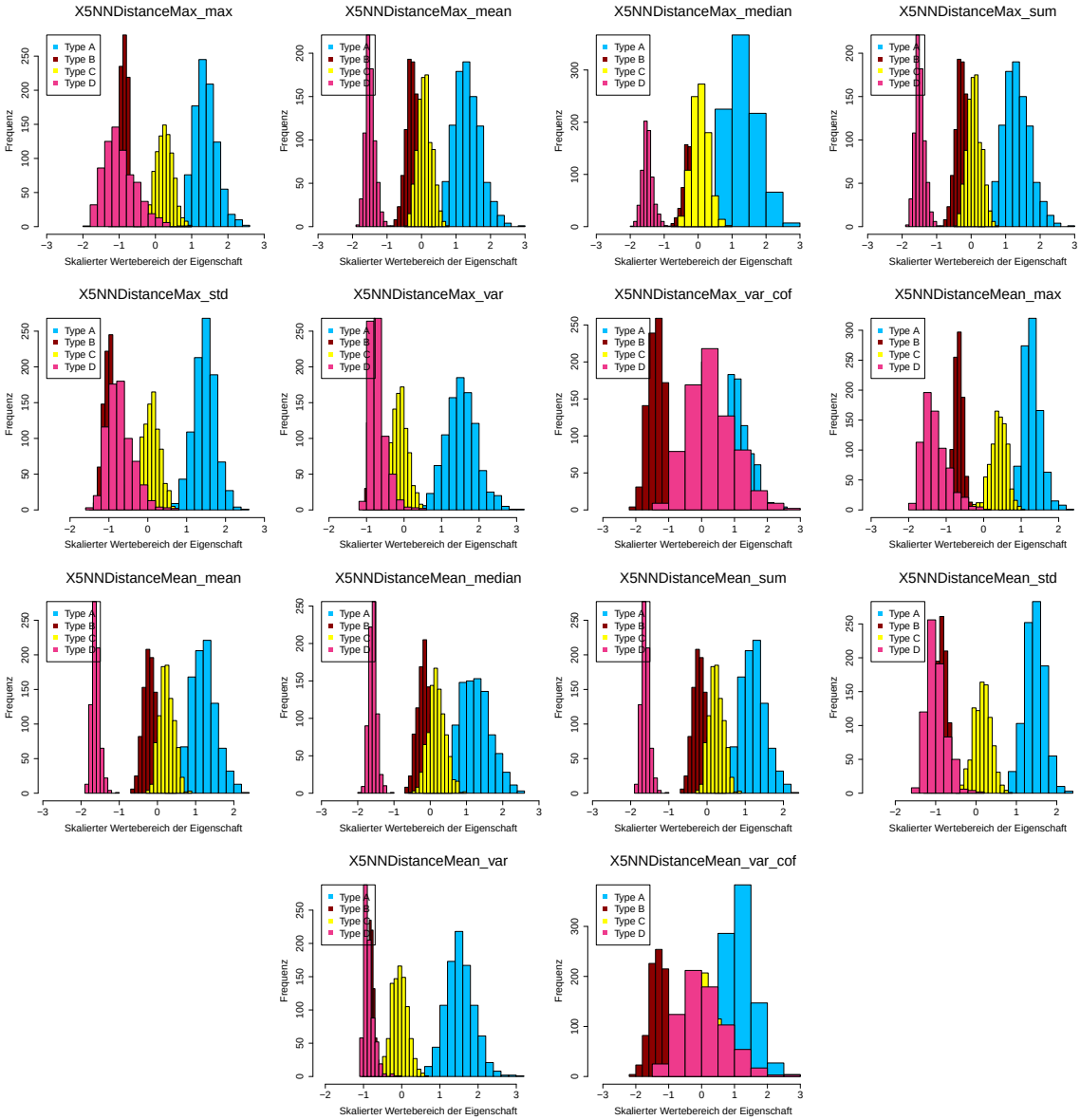

Abbildung A.12: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Nächsten-Nachbar-Eigenschaften Max und Mean mit der Nachbarschaft N mit $n = 5$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

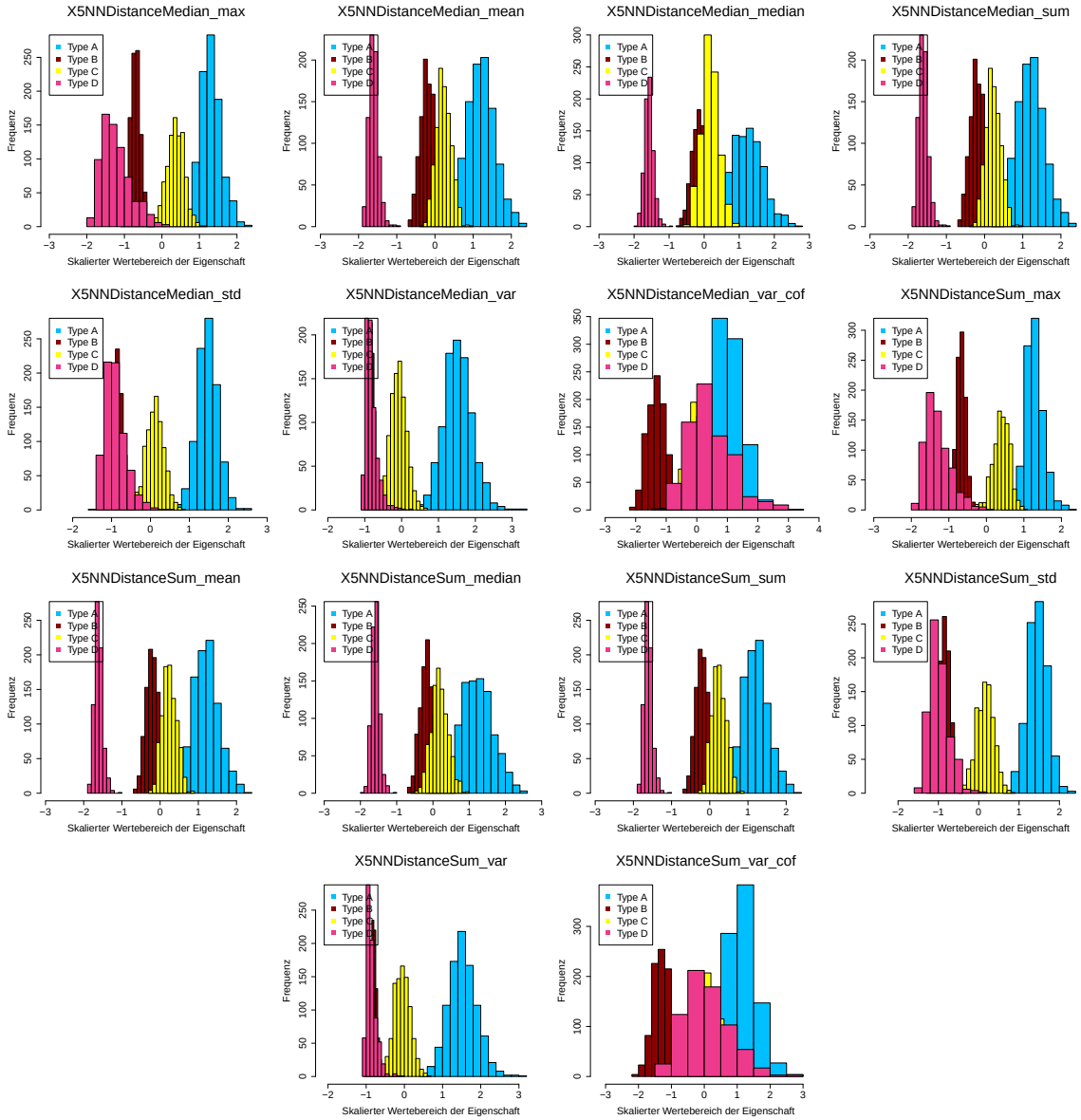

Abbildung A.13: Wertebereiche der die Beziehungen zwischen statischen und dynamischen Anfragen beschreibenden Nächsten-Nachbar-Eigenschaften Median und Summe mit der Nachbarschaft N mit $n = 5$ Nachbarn für die verschiedenen Typen von Probleminstanzen vorgestellt in Abschnitt 4.1.1

Anhang B

Problemstellungen und Lösungen, Selektion von Algorithmen

Im folgenden Abschnitt B.1 werden schwere und leichte Problemstellungen mithilfe von DVRP-Eigenschaften gegenübergestellt. Zusätzlich werden Interpretationshilfen in Form von Box-Plot-Verteilungsdiagrammen für die Visualisierungen der Gegenüberstellung angeboten. Abschnitt B.2 vervollständigt die in Unterabschnitt 6.1.2 betrachtete Rangverteilung bei der automatisierten Selektion von Algorithmen.

B.1 Trennung schwerer und leichter Problemstellungen

In den folgenden Abbildungen B.1, B.2, B.3 und B.4 sind 10% der besten und 10% der schlechtesten Lösungen der Algorithmen Spiral/Insert, Spiral8x8/Insert, NN 2-Opt/Insert und MMAS US/US mithilfe der Schwerpunkt-Eigenschaften (SE-Summe, SE-Median), der Flächen-Eigenschaft (FE) und der Distanz-Eigenschaft (DepotL-DOD) erfasst. Für die Interpretation der Diagramme sind zusätzlich die Box-Plot-Verteilungsdiagramme für die besten (grün) und schlechtesten (rot) Lösungen abgebildet. So soll eingeschätzt werden können, ob sich gute von schlechten Lösungen voneinander trennen lassen. Grundlage für die Visualisierungen bilden die in Unterabschnitt 5.2.2 verwendeten Daten.

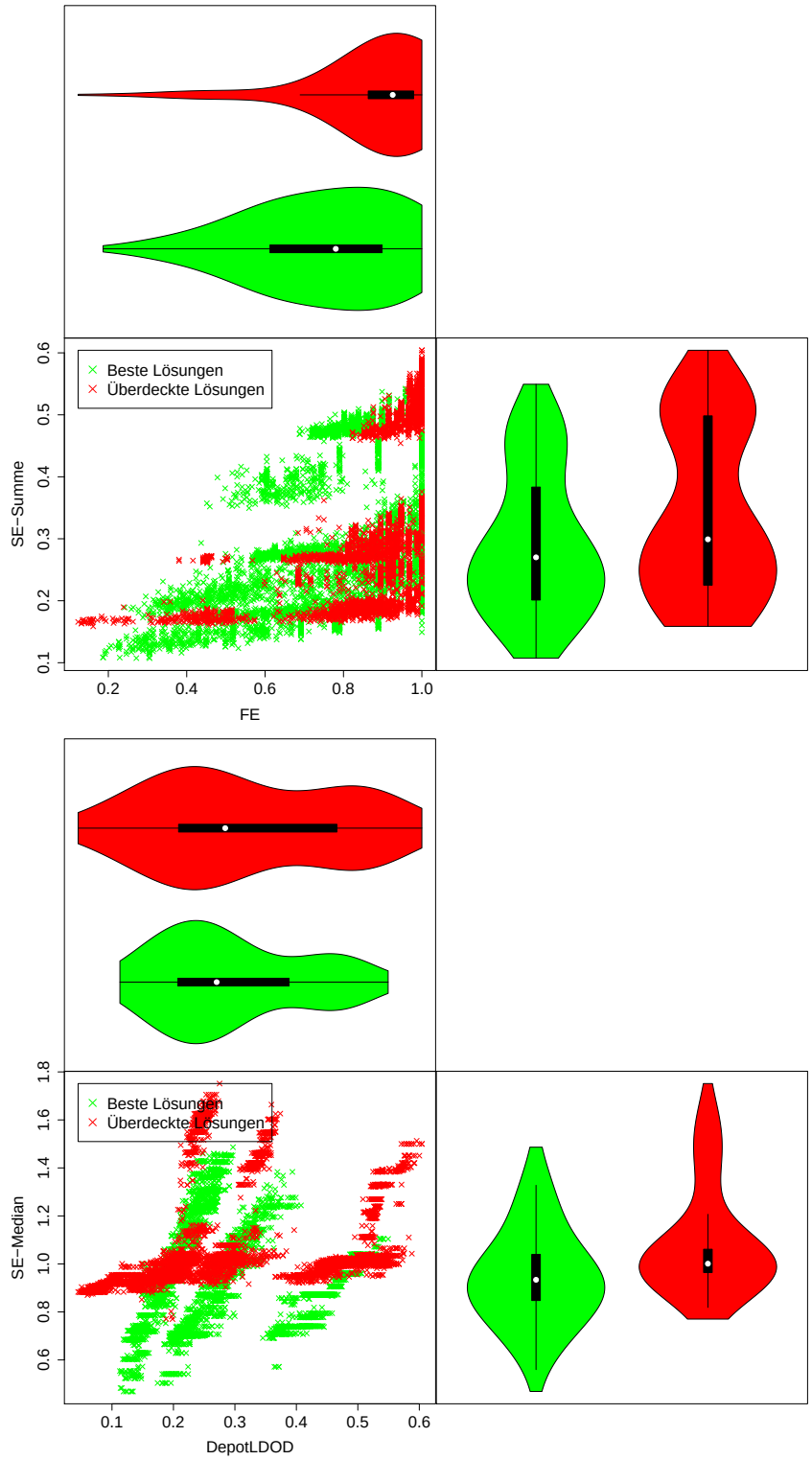

Abbildung B.1: Trennung schwerer und leichter Problemstellungen für den Spiral/Insert Algorithmus mithilfe der Schwerpunkt-Eigenschaften (SE-Summe, SE-Median), der Flächen-Eigenschaft (FE) und der Distanz-Eigenschaft (DepotLDOD)

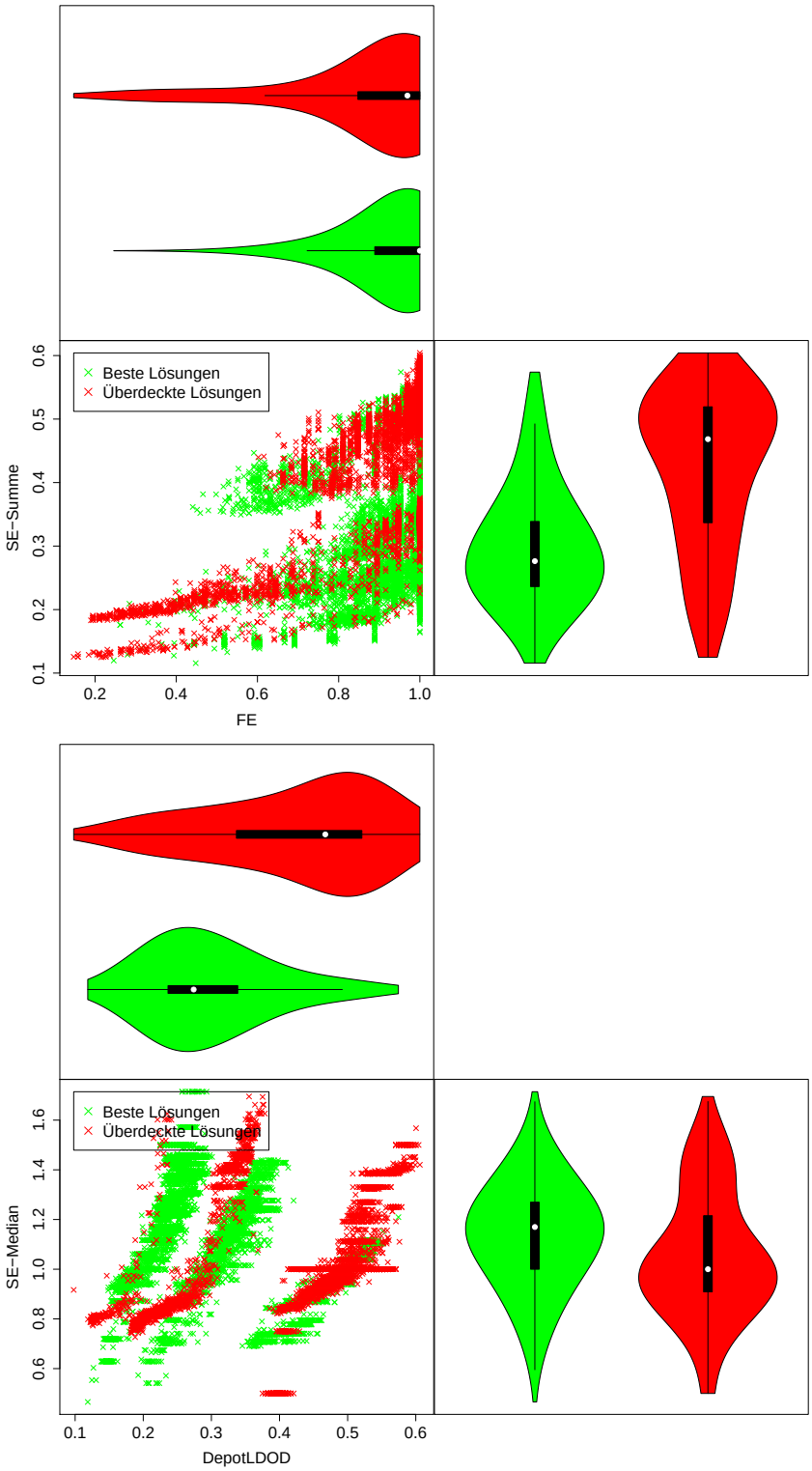

Abbildung B.2: Trennung schwerer und leichter Problemstellungen für den Spiral8x8/Insert Algorithmus mithilfe der Schwerpunkt-Eigenschaften (SE-Summe, SE-Median), der Flächen-Eigenschaft (FE) und der Distanz-Eigenschaft (DepotLDOD)

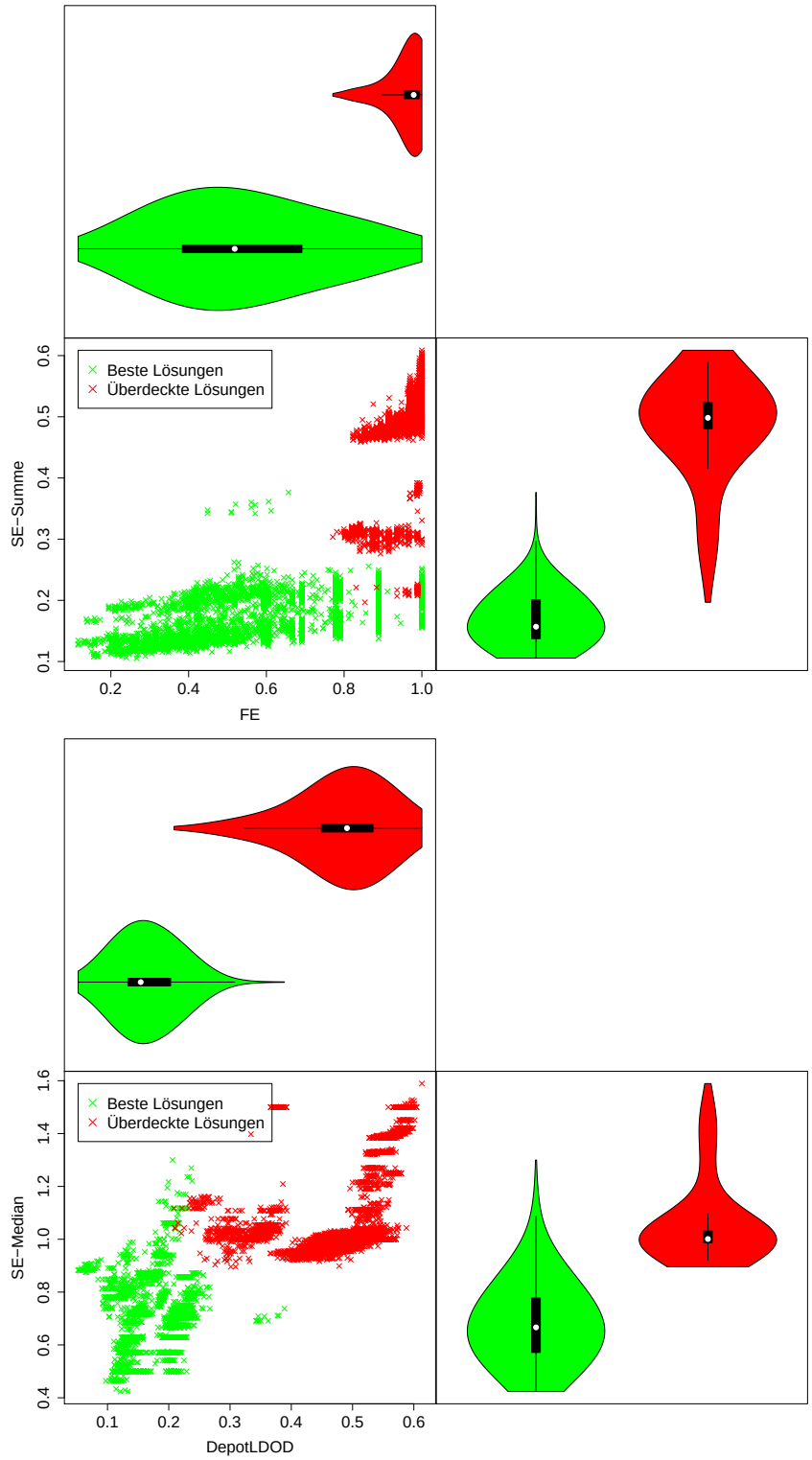

Abbildung B.3: Trennung schwerer und leichter Problemstellungen für den NN 2-Opt/Insert Algorithmus mithilfe der Schwerpunkt-Eigenschaften (SE-Summe, SE-Median), der Flächen-Eigenschaft (FE) und der Distanz-Eigenschaft (DepotLDOD)

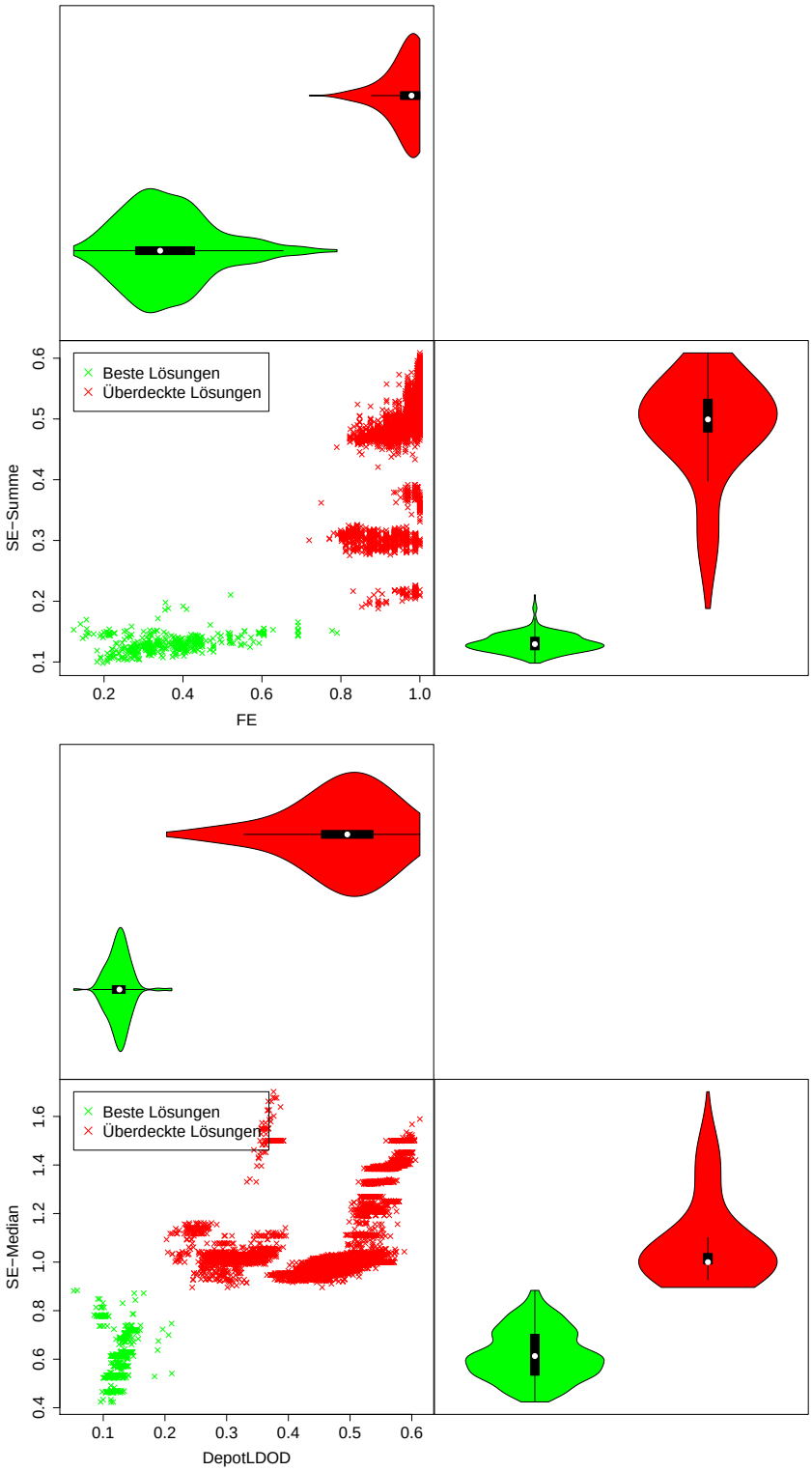

Abbildung B.4: Trennung schwerer und leichter Problemstellungen für den MMAS US/US Algorithmus mithilfe der Schwerpunkt-Eigenschaften (SE-Summe, SE-Median), der Flächen-Eigenschaft (FE) und der Distanz-Eigenschaft (DepotLDOD)

B.2 Rangverteilung der Algorithmen

In Unterabschnitt 6.1.2 wird unter anderem die Rangverteilung der betrachteten Algorithmen untersucht. Die Verteilung ist für die Algorithmen RM 8 Eigenschaften und MMASUS/US in Abbildung 6.4 abgebildet. Abbildung B.5 zeigt zusätzlich die Rangverteilungen für alle weiteren in Tabelle 6.5 betrachteten Algorithmen.

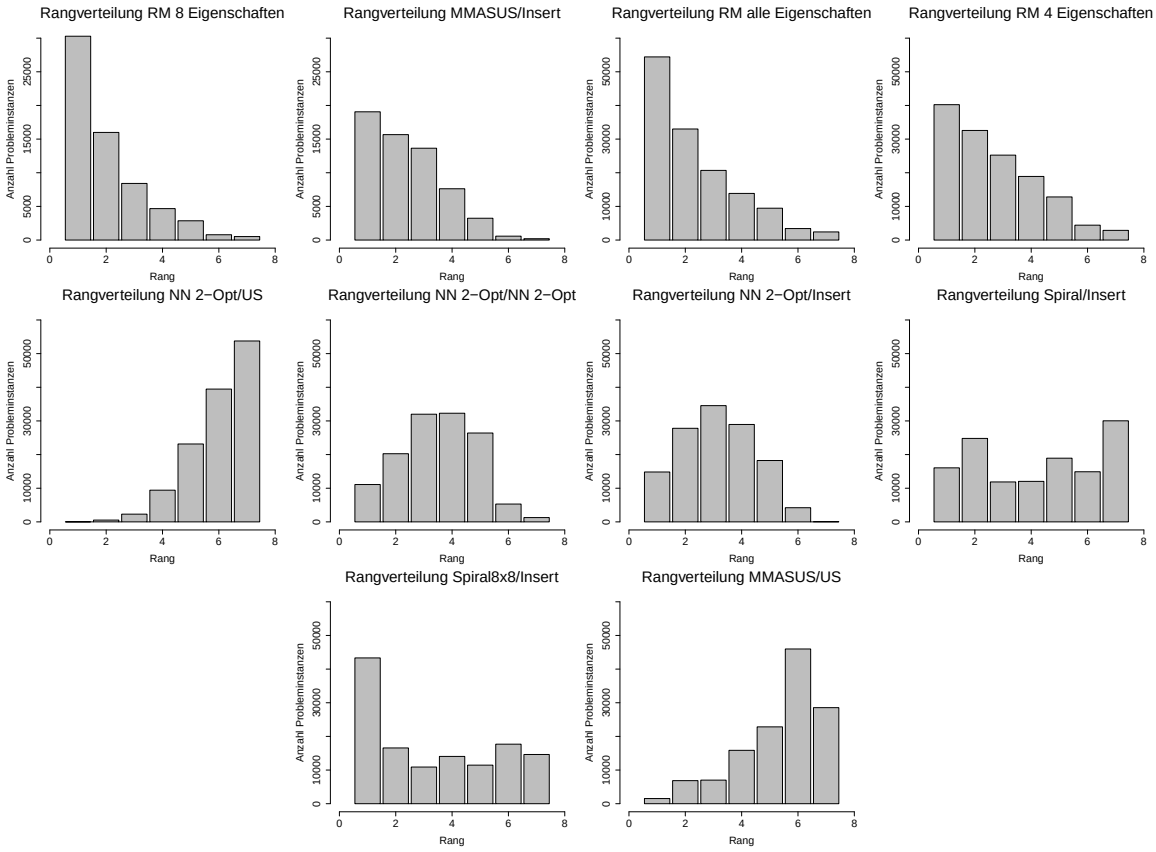

Abbildung B.5: Histogramme der Verteilung der Ränge der Algorithmen (der Rang entspricht der Platzierung der Lösung des Algorithmus)