Veronika Lesch

Self-Aware Optimization of Cyber-Physical Systems in Intelligent Transportation and Logistics Systems

Dissertation, Julius-Maximilians-Universität Würzburg
Fakultät für Mathematik und Informatik, 2021
Gutachter: Prof. Dr. Samuel Kounev, Julius-Maximilians-Universität Würzburg
Prof. Dr. Sven Tomforde, Christian-Albrechts-Universität zu Kiel

Datum der mündlichen Prüfung: 28.04.2022

Abstract

In today's world, circumstances, processes, and requirements for systems in general—in this thesis a special focus is given to the context of Cyber-Physical System (CPS)—are becoming increasingly complex and dynamic. In order to operate properly in such dynamic environments, systems must adapt to dynamic changes, which has led to the research area of Self-adaptive Systems (SASs). These systems can deal with changes in their environment and the system itself. In our daily lives, we come into contact with many different SASs that are designed to support and improve our way of life. In this work we focus on the two domains Intelligent Transportation Systems (ITS) and logistics as both domains provide complex and adaptable use cases to prototypical apply the contributions of this thesis. However, the contributions are not limited to these areas and can be generalized also to other domains such as the general area of CPS and Internet of Things (IoT) including smart grids or even intelligent computer networks. In ITS, real-time traffic control is an example adaptive system that monitors the environment, analyzes observations, and plans and executes adaptation actions. Another example is platooning, which is the ability of vehicles to drive with close inter-vehicle distances. This technology enables an increase in road throughput and safety, which directly addresses the increased infrastructure needs due to increased traffic on the roads. In logistics, the Vehicle Routing Problem (VRP) deals with the planning of road freight transport tours. To cope with the ever-increasing transport volume due to the rise of just-in-time production and online shopping, efficient and correct route planning for transports is important. Further, warehouses play a central role in any company's supply chain and contribute to the logistical success. The processes of storage assignment and order picking are the two main tasks in mezzanine warehouses highly affected by a dynamic environment. Usually, optimization algorithms are applied to find solutions in reasonable computation time. SASs can help address these dynamics by allowing systems to deal with changing demands and constraints.

For the application of SASs in the two areas ITS and logistics, the definition of adaptation planning strategies is the key success factor. A wide range of adaptation planning strategies for different domains can be found in the literature, and the operator must select the most promising strategy for the problem

at hand. However, the No-Free-Lunch theorem states that the performance of one strategy is not necessarily transferable to other problems. Accordingly, the algorithm selection problem, first defined in 1976, aims to find the best performing algorithm for the current problem. Since then, this problem has been explored more and more, and the machine learning community, for example, considers it a learning problem. The ideas surrounding the algorithm selection problem have been applied in various use cases, but little research has been done to generalize the approaches. Moreover, especially in the field of SASs, the selection of the most appropriate strategy depends on the current situation of the system. Techniques for identifying the situation of a system can be found in the literature, such as the use of rules or clustering techniques. This knowledge can then be used to improve the algorithm selection, or in the scope of this thesis, to improve the selection of adaptation planning strategies. In addition, knowledge about the current situation and the performance of strategies in similar previously observed situations provides another opportunity for improvements. This ongoing learning and reasoning about the system and its environment is found in the research area Self-aware Computing (SeAC).

In this thesis, we explore common characteristics of adaptation planning strategies in the domain of ITS and logistics presenting a self-aware optimization framework for adaptation planning strategies. We consider platooning coordination strategies from ITS and optimization techniques from logistics as adaptation planning strategies that can be exchanged during operation to better reflect the current situation. Further, we propose to integrate fairness and uncertainty handling mechanisms directly into the adaptation planning strategies. We then examine the complex structure of the logistics use cases VRP and mezzanine warehouses and identify their systems-of-systems structure. We propose a two-stage approach for vertical or nested systems and propose to consider the impact of intertwining horizontal or coexisting systems. More specifically, we summarize the six main contributions of this thesis as follows:

- First, we analyze specific characteristics of adaptation planning strategies with a particular focus on ITS and logistics. We use platooning and route planning in highly dynamic environments as representatives of ITS and we use the Rich Vehicle Routing Problem (rVRP) and mezzanine warehouses as representatives of the logistics domain. Using these case studies, we derive the need for situation-aware optimization of adaptation planning strategies and argue that fairness is an important consideration when applying these strategies in ITS. In logistics, we discuss that these complex systems can be considered as systems-of-systems and this struc-

ture affects each subsystem. Hence, we argue that the consideration of these characteristics is a crucial factor for the success of the system.

- Second, we design a self-aware optimization framework for adaptation planning strategies. The optimization framework is abstracted into a third layer above the application and its adaptation planning system, which allows the concept to be applied to a diverse set of use cases. Further, the Domain Data Model (DDM) used to configure the framework enables the operator to easily apply it by defining the available adaptation planning strategies, parameters to be optimized, and performance measures. The framework consists of four components: (i) Coordination, (ii) Situation Detection, (iii) Strategy Selection, and (iv) Parameter Optimization. While the coordination component receives observations and triggers the other components, the situation detection applies rules or clustering techniques to identify the current situation. The strategy selection uses this knowledge to select the most promising strategy for the current situation, and the parameter optimization applies optimization algorithms to tune the parameters of the strategy. Moreover, we apply the concepts of the SeAC domain and integrate learning and reasoning processes to enable ongoing advancement of the framework. We evaluate our framework using the platooning use case and consider platooning coordination strategies as the adaptation planning strategies to be selected and optimized. Our evaluation shows that the framework is able to select the most appropriate adaptation strategy and learn the situational behavior of the system.

- Third, we argue that fairness aspects, previously identified as an important characteristic of adaptation planning strategies, are best addressed directly as part of the strategies. Hence, focusing on platooning as an example use case, we propose a set of fairness mechanisms to balance positive and negative effects of platooning among all participants in a platoon. We design six vehicle sequence rotation mechanisms that continuously change the leader position among all participants, as this is the position with the least positive effects. We analyze these strategies on roads of different sizes and with different traffic volumes, and show that these mechanisms should also be chosen wisely.

- Fourth, we address the uncertainty characteristic of adaptation planning strategies and propose a methodology to account for uncertainty and also address it directly as part of the adaptation planning strategies. We address the use case of fueling planning along a route associated with highly

dynamic fuel prices and develop six utility functions that account for different aspects of route planning. Further, we incorporate uncertainty measures for dynamic fuel prices by adding penalties for longer travel times or greater distance to the next gas station. Through this approach, we are able to reduce the uncertainty at planning time and obtain a more robust route planning.

- Fifth, we analyze optimization of nested systems-of-systems for the use case rVRP. Before proposing an approach to deal with the complex structure of the problem, we analyze important constraints and objectives that need to be considered when formulating a real-world rVRP. Then, we propose a two-stage workflow to optimize both systems individually, flexibly, and interchangeably. We apply Genetic Algorithms (GAs) and Ant Colony Optimizations (ACOs) to both nested systems and compare the performance of our workflow with state-of-the-art optimization algorithms for this use case. In our evaluation, we show that the proposed two-stage workflow is able to handle the complex structure of the problem and consider all real-world constraints and objectives.
- Finally, we study coexisting systems-of-systems by optimizing typical processes in mezzanine warehouses. We first define which ergonomic and economic constraints and objectives must be considered when addressing a real-world problem. Then, we analyze the interrelatedness of the storage assignment and order picking problems; we identify opportunities to design optimization approaches that optimize all objectives and aim for a good overall system performance, taking into account the interdependence of both systems. We use the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for storage assignment and ACO for order picking and adapt them to the specific requirements of horizontal systems-of-systems. In our evaluation, we compare our approaches to state-of-the-art approaches in mezzanine warehouses and show that our proposed approaches increase the system performance.

Our proposed approaches provide important contributions to both academic research and practical applications. To the best of our knowledge, we are the first to design a self-aware optimization framework for adaptation planning strategies that integrates situation-awareness, algorithm selection, parameter tuning, as well as learning and reasoning. Our evaluation of platooning coordination shows promising results for the application of the framework. Moreover, our proposed strategies to compensate for negative effects of platooning represent an important milestone, which could lead to higher acceptance of this

technology in society and support its future adoption in the real world. The proposed methodology and utility functions that address uncertainty are an important step to improving the capabilities of SAS in an increasingly turbulent environment. Similarly, our contributions to systems-of-systems optimization are major contributions to the state of logistics and systems-of-systems research. Finally, we select real-world use cases for the application of our approaches and cooperate with industrial partners, which highlights the practical relevance of our contributions. The reduction of manual effort and required expert knowledge in our self-aware optimization framework is a milestone in bridging the gap between academia and practice. One of our partners integrated the two-stage approach to tackling the rVRP into its software system, improving both time to solution and solution quality. In conclusion, the contributions of this thesis have spawned several research projects such as a long-term industrial project on optimizing tours and routes in parcel delivery funded by *Bayerisches Verbundforschungsprogramm (BayVFP) – Digitalisierung* and further collaborations, opening up many promising avenues for future research.

Deutsche Zusammenfassung

In der heutigen Welt werden die Umstände, Prozesse und Anforderungen an Systeme im allgemeinen—in dieser Arbeit wird der Fokus besonders auf cyber-physische Systeme (engl. CPS) gelegt—immer komplexer. Um in solch dynamischen Umgebungen ordnungsgemäß zu funktionieren, müssen sich diese Systeme an Veränderungen anpassen. Diese Herausforderungen führten zu der Entstehung des Forschungsbereichs selbst-adaptiver Systeme (engl. Self-adaptive System (SAS)). Diese Systeme können mit Veränderungen in ihrer Umgebung, als auch in sich selbst umgehen und sich an geänderte Gegebenheiten anpassen. In unserem alltäglichen Leben kommen wir daher zunehmend mit SAS in Berührung, welche unsere Lebensqualität unterstützen und verbessern sollen. In dieser Arbeit konzentrieren wir uns auf die beiden Bereiche Intelligenter Transportsysteme (engl. Intelligent Transportation Systems (ITS)) und Logistik, da beide Bereiche komplexe und anpassbare Anwendungsfälle bieten, um die Beiträge dieser Arbeit prototypisch anzuwenden. Die vorgestellten Beiträge sind jedoch nicht auf diese Bereiche beschränkt und können auch auf andere Bereiche wie den allgemeinen Bereich von cyber-physischen Systemen und dem Internet der Dinge einschließlich intelligenter Stromnetze oder intelligenter Computernetze verallgemeinert werden. Als ein Beispiel für adaptive Systeme in der realen Welt kann die Echtzeit-Verkehrssteuerung genannt werden. Diese Systeme überwachen die Umgebung, analysieren Beobachtungen und planen Anpassungsmaßnahmen um den Verkehrsfluss zu regulieren. Ein weiteres Beispiel ist das sogenannte Platooning, welches die Fähigkeit beschreibt, in Gruppen mit geringen Abständen zwischen den beteiligten Fahrzeugen zu fahren. Das stetig wachsende Verkehrsvolumen auf den Straßen führt automatisch zu einem erhöhten Infrastrukturbedarf, den Behörden derzeit durch Neubau und Erweiterung der existierenden Infrastruktur begegnen. Platooning ist eine vielversprechende Technologie, die durch die Reduzierung der Mindestabstände von Fahrzeugen und Kommunikation zwischen diesen, automatisch zu einer Erhöhung des Verkehrsdurchsatzes sowie der Sicherheit auf der verwendeten Infrastruktur führt. Auch im Bereich der Logistik finden sich diverse Beispiele für SAS, wie zum Beispiel die Tourenplanung (engl. Vehicle Routing Problem (VRP)) welches sich mit der Planung von Touren für den Straßengütervekehr befasst. Zur Bewältigung

des ständig steigenden Transportaufkommens aufgrund zunehmender Just-in-Time-Produktion und erhöhter Nachfrage durch Online-Shopping ist eine effiziente und korrekte Routenplanung für Warentransporte besonders wichtig. Üblicherweise werden Optimierungsalgorithmen angewandt, um sinnvolle Lösungen in angemessener Rechenzeit zu finden. Durch die Anwendung von Konzepten der SAS kann die Dynamik des Problems berücksichtigt werden, indem es den Umgang mit sich ändernden Anforderungen, Einschränkungen und spontan eingehenden Aufträgen ermöglicht. Darüber hinaus spielen Lagerhäuser eine zentrale Rolle in der Lieferkette von Unternehmen und tragen maßgeblich zum logistischen Erfolg bei. Fachbodenregallager (engl. mezzanine warehouses) sind laut Expertenschätzung die am häufigsten verwendeten Lager wenn Mitarbeiter die eingelagerten Güter manuell ein- und auslagern (engl. picker-to-part). Die Prozesse der Lagerzuweisung und der Kommissionierung sind die beiden Hauptaufgaben in Fachbodenregallagern, welche ebenfalls häufig mit Optimierungsalgorithmen gelöst werden. Beide Prozesse müssen in einer dynamischen Umgebung ablaufen, für die SAS einen vielversprechenden Lösungsansatz darstellt.

Für die Anwendung von SAS in diesen beiden Bereichen ist die Definition von Anpassungsplanungsstrategien (engl. adaptation planning strategies) der Schlüsselfaktor für den Erfolg des Gesamtsystems. In der Literatur finden sich zahlreiche Anpassungsplanungsstrategien für verschiedene Anwendungsbereiche, was dazu führt, dass der Anwender die vielversprechendste Strategie für das jeweilige Problem auswählen muss. Das No-Free-Lunch-Theorem besagt jedoch, dass die Leistung einer Strategie nicht direkt auf andere Probleme übertragbar ist. Dementsprechend zielt das 1976 erstmals definierte Problem der Algorithmenauswahl darauf ab, den leistungsfähigsten Algorithmus für das aktuelle Problem zu finden. Seitdem wurde diese Problemstellung immer weiter erforscht, und wird beispielsweise von der Forschungsgemeinschaft, welche sich mit maschinellem Lernen beschäftigt, als Lernproblem angesehen. In der Literatur lassen sich vielfältige Ideen finden, welche auf die Algorithmentauswahl in verschiedenen Anwendungsfällen abzielen. Jedoch wurde bisher nur wenig Forschung betrieben, um diese Ansätze zu verallgemeinern und auf andere Anwendungsfälle zu übertragen. Darüber hinaus hängt besonders im Bereich der SAS die Auswahl der am besten geeigneten Strategie von der aktuellen Situation des Systems ab. In der Literatur finden sich Techniken zur Identifizierung der Situation eines Systems, z. B. durch Anwendung von Regeln oder Clustering-Techniken. Dieses Wissen kann dann verwendet werden, um die Auswahl der Algorithmen, oder im Rahmen dieser Arbeit, die Auswahl der Anpassungsplanungsstrategien zu verfeinern. Darüber hinaus bietet die Kenntnis über die aktuelle Situation und die Leistung von Anpassungsplanungsstrategien in ähnlichen, zuvor beobachteten Situationen eine weitere Möglichkeit für Verbesserungen. Dieses Lernen und Nachdenken (engl. reasoning) über das System und seine Umgebung ist Kernbestandteil des Forschungsbereichs der sich selbst bewusster Computer Systeme (engl. SeAC).

In dieser Arbeit untersuchen wir gemeinsame Merkmale von Anpassungsplanungsstrategien in den Anwendungsbereichen ITS sowie Logistik und stellen ein sich seiner selbst bewusstes (engl. self-aware) Rahmenkonzept zur Optimierung dieser Strategien vor. Wir betrachten Platooning-Koordinations-strategien aus dem Bereich ITS und Optimierungstechniken aus der Logistik als Anpassungsplanungsstrategien, die unter Berücksichtigung der aktuellen Situation ausgetauscht und optimiert werden können. Darüber hinaus schlagen wir vor, die Aspekte Fairness und Unsicherheit direkt in solche Strategien zu integrieren. Anschließend untersuchen wir die komplexe Struktur der logistischen Anwendungsfälle VRP und Fachbodenregallager und identifizieren ihre Struktur bestehend aus Systemen von Systemen (engl. System-of-Systems). Wir entwerfen einen zweistufigen Ansatz für vertikale oder verschachtelte Systeme und schlagen vor, die Auswirkungen der Verflechtung horizontaler oder koexistierender Systeme zu berücksichtigen. Im Einzelnen fassen wir die sechs Hauptbeiträge dieser Arbeit wie folgt zusammen:

- Zu Beginn analysieren wir die spezifischen Merkmale von Anpassungsplanungsstrategien mit besonderem Augenmerk auf ITS und Logistik. Wir verwenden Platooning und Routenplanung in hochdynamischen Umgebungen als Repräsentanten für ITS und rVRP sowie Fachbodenregallager als Vertreter der Logistikdomäne. Anhand dieser Fallstudien leiten wir die Notwendigkeit einer situationsgerechten Optimierung von Anpassungsplanungsstrategien ab und argumentieren, dass Fairness ein wichtiger Aspekt bei der Anwendung dieser Strategien in ITS ist. Im Bereich der Logistik erörtern wir, dass diese komplexen Systeme als System von Systemen betrachtet werden können und dass diese Struktur die Leistung der einzelnen Teilsysteme beeinflusst. Daher argumentieren wir, dass die Berücksichtigung dieser Merkmale ein entscheidender Faktor für den Erfolg des Gesamtsystems ist.

- Zweitens entwerfen wir ein sich seiner selbst bewusstes Rahmenwerk zur Optimierung von Anpassungsplanungsstrategien. Wir abstrahieren den Rahmen für die Optimierung der Strategien auf eine dritte Ebene oberhalb des Anwendungsfalls und seinem Anpassungsplanungssystems, was die Übertragung der Konzepte auf eine Vielzahl von Anwendungsfällen ermöglicht. Darüber hinaus schlagen wir ein Domänendatenmodell (engl. DDM) für die Konfiguration des Rahmenwerks vor, so dass der Anwender durch die Definition von verfügbaren Anpassungsplanungsstrategien, der zu optimierenden Parameter und der Leistungsmaße das Rahmenwerk individuell anwenden kann. Der Rahmen besteht aus vier Komponenten: (i) Koordination, (ii) Situationserkennung, (iii) Strategieauswahl, und (iv) Parameteroptimierung. Während die Koordinationskomponente Beobachtungen empfängt und die anderen Komponenten aufruft, wendet die Situationserkennung Regeln oder Clustering-Techniken an, um die aktuelle Situation zu identifizieren. Die Strategieauswahl nutzt dieses Wissen, um die vielversprechendste Strategie für die aktuelle Situation auszuwählen, und die Parameteroptimierung wiederum wendet Optimierungsalgorithmen an, um die Parameter der Strategie einzustellen. Darüber hinaus wenden wir die Konzepte aus dem Bereich des SeAC an und integrieren Schlussfolgerungs- und Lernprozesse, um eine kontinuierliche Weiterentwicklung des Rahmenwerks zu ermöglichen. Wir evaluieren unser Rahmenwerk anhand des Anwendungsfalls Platooning und wählen dynamisch Platooning-Koordinationsstrategien aus und optimieren deren Parameterbelegung. Unsere Evaluation zeigt, dass das Rahmenwerk in der Lage ist, die am besten geeignete Anpassungsstrategie auszuwählen und das situative Verhalten des Systems zu erlernen.

- Drittens argumentieren wir, dass der Aspekt Fairness am besten direkt in den Strategien berücksichtigt werden sollte. Daher schlagen wir eine Reihe von Fairness-Mechanismen vor, um positive und negative Auswirkungen des Platooning zwischen allen Teilnehmern eines Platoons auszugleichen. Wir entwerfen sechs Mechanismen zur Rotation der Fahrzeugreihenfolge im Platoon, die die Führungsposition unter allen Teilnehmern kontinuierlich wechseln, da dies die Position mit den geringsten positiven Auswirkungen ist. Wir analysieren diese Strategien auf Straßen unterschiedlicher Größe sowie mit unterschiedlichem Verkehrsaufkommen und zeigen, dass auch diese Mechanismen mit Bedacht gewählt werden sollten.

- Viertens befassen wir uns mit Unsicherheit und schlagen eine Methodik vor, um Unsicherheit zu berücksichtigen und sie auch direkt in den Anpassungsplanungsstrategien zu behandeln. Wir befassen uns mit der optimierten Auswahl von Tankstellen entlang einer Route, die mit hochdynamischen Treibstoffpreisen einhergeht, und entwickeln sechs Nutzenfunktionen, welche verschiedene Aspekte der Routenplanung berücksichtigen. Darüber hinaus integrieren wir Unsicherheitsmaße für dynamische Kraftstoffpreise, indem wir Strafen für längere Fahrtzeiten oder eine größere Entfernung zur nächsten Tankstelle hinzufügen. Durch diesen Ansatz sind wir in der Lage, die Unsicherheit zum Planungszeitpunkt zu reduzieren und erhalten so eine robustere Routenplanung.

- Fünftens erforschen wir die Optimierung von verschachtelten System von Systemen für den Anwendungsfall rVRP. Bevor wir einen Ansatz zur Bewältigung der komplexen Struktur des Problems vorschlagen, analysieren wir wichtige Einschränkungen und Ziele, die bei der Erstellung eines realen rVRP berücksichtigt werden sollten. Dann schlagen wir einen zweistufigen Arbeitsablauf (engl. Workflow) vor, mit dem beide Systeme individuell, flexibel und austauschbar optimiert werden können. Wir wenden einen genetischen Algorithmus (engl. GA) und einen Ameisenalgorithmus (engl. ACO) auf beide Systeme an und vergleichen die Leistung unseres Arbeitsablaufs mit weit verbreiteten Optimierungsalgorithmen für diesen Anwendungsfall. In unserer Bewertung zeigen wir, dass der vorgeschlagene zweistufige Arbeitsablauf in der Lage ist, die komplexe Struktur der Problemstellung zu bewältigen und alle realitätsnahen Einschränkungen und Ziele zu berücksichtigen.

- Schließlich untersuchen wir koexistierende Systeme von Systemen, indem wir typische Prozesse in Fachbodenregallager optimieren. Zunächst definieren wir, welche ergonomischen und wirtschaftlichen Randbedingungen sowie Ziele bei der Erstellung eines realen Problems berücksichtigt werden müssen. Dann analysieren wir die Wechselbeziehung zwischen Lagerzuordnungs- und Kommissionierproblemen und zeigen Möglichkeiten auf, Optimierungsansätze zu entwerfen, die alle Ziele optimieren sowie eine gute Gesamtsystemleistung anstreben, während gleichzeitig die gegenseitige Abhängigkeit beider Systeme berücksichtigt wird. Wir verwenden NSGA-II für die Lagerbelegung und ACO für die Kommissionierung und passen beide an die spezifischen Anforderungen horizontaler Systeme von Systemen an. In unserer Evaluierung vergleichen wir unsere Ansätze mit State-of-the-Art-Ansätzen in Fachbodenregallagern und zeigen, dass die von uns vorgeschlagenen Ansätze die Systemleistung erhöhen.

Die von uns vorgeschlagenen Ansätze liefern sowohl für die akademische Forschung, als auch für praktische Anwendungen wichtige Beiträge. Wir sind, unseres Wissens nach, die Ersten, die ein sich seiner Selbst bewusstes Rahmenwerk zur Optimierung von Anpassungsplanungsstrategien entwerfen, das Situationsbewusstsein, Algorithmenauswahl, Parameteroptimierung sowie Lernen und Schlussfolgern integriert. Unsere Evaluierung der Platooning-Koordination zeigt vielversprechende Ergebnisse für die Anwendung des Rahmenwerks, da es in der Lage ist, die am besten geeignete Anpassungsplanungsstrategie auszuwählen und situatives Verhalten zu erlernen. Darüber hinaus stellen die von uns vorgeschlagenen Strategien zur Kompensation negativer Auswirkungen des Platooning einen wichtigen Meilenstein für die weitere Forschung im Bereich der Platooning-Koordination dar, was zu einer höheren Akzeptanz in der Gesellschaft und einer wahrscheinlicheren Übernahme der Technologie in der realen Welt führen könnte. Die vorgeschlagene Methodik und die Nutzenfunktionen, die sich mit Unsicherheit befassen, sind ein wichtiger Schritt zur Verbesserung der Fähigkeiten von SAS in einer zunehmend turbulenten Umgebung. In ähnlicher Weise sind unsere Beiträge zur Optimierung von Systemen bestehend aus weiteren Systemen wichtige Beiträge zum Stand der Logistik- und System-of-System-Forschung. Schlussendlich haben wir für die Anwendung unserer Ansätze reale Anwendungsfälle ausgewählt und mit Industriepartnern zusammengearbeitet, was die praktische Relevanz unserer Beiträge unterstreicht. Die Reduzierung des manuellen Aufwands und des erforderlichen Expertenwissens in unserem selbstlernenden Rahmenwerk ist ein Meilenstein in der Überbrückung der Kluft zwischen Wissenschaft und Praxis. Einer unserer Partner hat den zweistufigen Ansatz zur Bewältigung des rVRP bereits in sein Softwaresystem integriert, was sowohl die Zeit zur Lösungsfindung als auch die Lösungsqualität verbessert. Zusammenfassend führten die Beiträge dieser Arbeit zu weiteren Forschungsprojekten, zum Beispiel einem Industrie-Projekt zur Optimierung von Paketzustellungen gefördert vom *Bayerischer Verbundforschungsprogramm (BayVFP) - Digitalisierung*, sowie weiteren Kooperationen, die vielversprechende Perspektiven für die künftige Forschung eröffnen.

Acknowledgements

This thesis would not have been possible without the inspiration, support, and guidance of a significant number of people. I want to thank every one of them.

First of all, I would like to thank Prof. Dr.-Ing. Samuel Kounev, who inspired me to start my Ph.D. I am grateful for the opportunity to be part of his Descartes Research Group, for his constant trust and support, as well as the freedom to pursue my interests. I would also like to thank Prof. Dr.-Ing. Sven Tomforde for being my second reviewer and for his openness to all my questions.

I would like to thank my former and current colleagues with whom I had the pleasure of working with throughout the years, namely Dr. Simon Spinner, Dr. Nikolas Herbst, Dr. Jürgen Walter, Dr. Jóakim von Kistowski, Dr. Lukas Iffländer, Dr. André Bauer, Dr. Marwin Züfle, Dr. Johannes Grohmann, Dr. Norbert Schmitt, Lukas Beierlieb, Vanessa Borst, Bohdan Dovhan, Simon Eismann, Marius Hadry, Stefan Herrnleben, Dennis Kaiser, Robert Leppich, Maximilian Meißner, Thomas Prantl, Maximilian Schwinger, Florian Spieß, and Martin Sträßer. Additionally, I would like to thank all my student and research assistants who supported my research in diverse work-packages. Further, I would like to thank Susanne Stenglin, Erika Littmann, and Fritz Kleemann for their administrative support at the Chair of Computer Science II.

Especially, I would like to thank Jun.-Prof. Dr. Christian Krupitzer for joining the Descartes Research Group as Post-Doc, for always having an open ear, understanding, and good advices for me. Through his support I have made it through the difficult times of the Ph.D. for which I am very grateful. Further, I would like to express my special thanks to my dearest colleagues and friends Dr. Marwin Züfle and Stefan Herrnleben, who have accompanied and supported me since the beginning of my computer science studies. You have enriched my studies and Ph.D. time as best friends and gave me the confidence and courage to pursue my goals. Last, I would like to thank my colleagues from the Kaffeekränzchen group. Together we have spent difficult times of the pandemic with many hours of laughter and fruitful discussions.

Further, I would like to thank all of my co-authors that are not part of the Descartes Research Group, of which I can only name a few: Prof. Dr.-Ing. Sergio Montenegro, Prof. Dr. Christian Becker, Prof. Dr. Michele Segata, Jun.-Prof. Dr. Anthony Stein, Martin Breitbach, Johannes Hefter, Elia Henrichs, Nico Keil, Maximilian König, Tanja Noack, and Kevin Stubenrauch. Additionally, I would also like to thank all my industry partners for the good cooperation as well as the insights into practice that were granted.

I would like to thank my family for always being there for me. To my parents Frank and Gudrun for giving me the opportunity to follow my path and achieve my dreams. To my sister Elisabeth for always believing in me.

Last but not least, to the love of my life Tobias for your unconditional love, for being my inspiration, for the heated while fruitful discussions, and for always being by my side. Without you by my side, I would not be in this place today.

Contents

Publication List

Peer-Reviewed Journal Articles

[LKK^{+}21a] Tackling the Rich Vehicle Routing Problem with Nature-Inspired Algorithms; Veronika Lesch, Maximilian König, Samuel Kounev, Anthony Stein, Christian Krupitzer; in Applied Intelligence; Springer; 2021; **Accepted - In Press, Impact Factor (2020): 5.086**.

[LBS^{+}21] An Overview on Approaches for Coordination of Platoons; Veronika Lesch, Martin Breitbach, Michele Segata, Christian Becker, Samuel Kounev, Christian Krupitzer; in IEEE Transactions on Intelligent Transportation Systems; 2021; **Impact Factor (2020): 6.492**.

[LKS^{+}21] A Comparison of Mechanisms for Compensating Negative Impacts of System Integration; Veronika Lesch, Christian Krupitzer, Kevin Stubenrauch, Nico Keil, Christian Becker, Samuel Kounev, Michele Segata; in Future Generation Computer Systems (2021); 116 117–131. **Impact Factor (2020): 7.187**.

[ZML^{+}21] A Machine Learning-based Workflow for Automatic Detection of Anomalies in Machine Tools; Marwin Züfle, Felix Moog, Veronika Lesch, Christian Krupitzer, Samuel Kounev; in ISA Transactions; Elsevier; 2021; **Impact Factor (2020): 5.468**.

[KLR^{+}20] Towards Self-Aware Multirotor Formations; Dennis Kaiser, Veronika Lesch, Julian Rothe, Michael Strohmeier, Florian Spiess, Christian Krupitzer, Sergio Montenegro, Samuel Kounev; in Computers (2020); 9(7).

Journal Articles Under Review or in Preparation

[LHKK21c] A Self-Aware Optimization Framework for Adaptation Planning Strategies; Veronika Lesch, Marius Hadry, Christian Krupitzer, Samuel Kounev; in ACM TAAS; 2021 **Under Review, Impact Factor (2022): 2.47**.

[LMK$^+$21a] Optimizing Storage Assignment, Order Picking, and their Interaction in Mezzanine Warehouses; Veronika Lesch, Patrick B. M. Müller, Moritz Krämer, Samuel Kounev, Christian Krupitzer; in Applied Intelligence; Springer; 2021; **Under Review, Impact Factor (2020): 5.086**.

[HLS$^+$21] A Literature Review on Optimization Techniques for Adaptation Planning in Adaptive Systems: State of the Art and Research Directions; Elia Henrichs, Veronika Lesch, Martin Straesser, Samuel Kounev, Christian Krupitzer; in Information and Software Technology; Elsevier; 2021; **Under Review (Minor Revision), Impact Factor (2020): 2.73**.

Peer-Reviewed Conference Papers

[LNH$^+$21] Towards Situation-Aware Meta-Optimization of Adaptation Planning Strategies; Veronika Lesch, Tanja Noack, Johannes Hefter, Samuel Kounev, Christian Krupitzer; Proceedings of the 2nd IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS 2021) **Best Paper Candidate**.

[LBHK18] FOX: Cost-Awareness for Autonomic Resource Management in Public Clouds; Veronika Lesch, André Bauer, Nikolas Herbst, Samuel Kounev; in Proceedings of the 9th ACM/SPEC International Conference on Performance Engineering (ICPE 2018) (2018).

[HLL$^+$21] ComBench: A Benchmarking Framework for Publish/Subscribe Communication Protocols under Network Limitations; Stefan Herrnleben, Maximilian Leidinger, Veronika Lesch, Thomas Prantl, Johannes Grohmann, Christian Krupitzer, Samuel Kounev; in Proceedings of the 14th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS); 2021.

[HGR$^+$21] A Simulation-based Optimization Framework for Online Adaptation of Networks; Stefan Herrnleben, Johannes Grohmann, Pitor Rygielski, Veronika Lesch, Christian Krupitzer, Samuel Kounev; in Proceedings of the 12th EAI International Conference on Simulation Tools and Techniques (SIMUtools), H. Song, D. Jiang (Hrsg.) (2021); 513–532.

[ISR$^+$19] Performance Oriented Dynamic Bypassing for Intrusion Detection Systems; Lukas Iffländer, Jonathan Stoll, Nishant Rawtani, Veronika Lesch, Klaus-Dieter Lange, Samuel Kounev; in Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (2019); 159–166.

[BLV$^+$19] Chamulteon: Coordinated Auto-Scaling of Micro-Services; André Bauer, Veronika Lesch, Laurens Versluis, Alexey Ilyushkin, Nikolas Herbst, Samuel Kounev; in Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS) (2019).

[ZBL$^+$19] Autonomic Forecasting Method Selection: Examination and Ways Ahead; Marwin Züfle, André Bauer, Veronika Lesch, Christian Krupitzer, Nikolas Herbst, Samuel Kounev, Valentin Curtef; in Proceedings of the 16th IEEE International Conference on Autonomic Computing (ICAC) (2019).

Peer-Reviewed Workshop Papers

[LHKK21b] Utility-based Vehicle Routing Integrating User Preferences; Veronika Lesch, Marius Hadry, Samuel Kounev, Christian Krupitzer; in Proceedings of 3rd International Workshop on Pervasive Computing for Vehicular Systems (PerVehicle), 2021 (2021).

[LKT19b] Multi-objective Optimisation in Hybrid Collaborating Adaptive Systems; Veronika Lesch, Christian Krupitzer, Sven Tomforde; in Proceedings of the 7th edition in the Series on Autonomously Learning and Optimising Systems (SAOS), co-located with 32nd GI/ITG ARCS 2019 (2019).

[LKT19a] Emerging Self-Integration through Coordination of Autonomous Adaptive Systems; Veronika Lesch, Christian Krupitzer, Sven Tomforde; in Proceedings of the 4th IEEE International Workshops on Foundations and Applications of Self* Systems, FAS*W@ICAC/SASO 2019 (2019).

[KLP$^+$19] A Modular Simulation Framework for Analyzing Platooning Coordination; Christian Krupitzer, Veronika Lesch, Martin Pfannemüller, Christian Becker, Michele Segata; in Proceedings of the 1st ACM Workshop on Technologies, mOdels, and Protocols for Cooperative Connected Cars (TOP-Cars), Colocated with ACM MobiHoc 2019 (2019).

Book Chapters

[Les20] Toward a Framework for Self-Learning Adaptation Planning through Optimization; Veronika Lesch; in Organic Computing: Doctoral Dissertation Colloquium 2020 (2020).

Technical Reports

[LKK+21b] A Case Study of Vehicle Route Optimization; Veronika Lesch, Maximilian König, Samuel Kounev, Anthony Stein, Christian Krupitzer; (2021).

[LHKK21a] A Case Study on Optimization of Platooning Coordination; Veronika Lesch, Marius Hadry, Samuel Kounev, Christian Krupitzer; (2021).

[LMK+21b] A Case Study on Optimization of Warehouses; Veronika Lesch, Patrick B. M. Müller, Moritz Krämer, Samuel Kounev, Christian Krupitzer; (2021).

[KML+20] A Survey on Human Machine Interaction in Industry 4.0. Christian Krupitzer, Sebastian Müller, Veronika Lesch, Marwin Züfle, Janick Edinger, Alexander Lemken, Dominik Schäfer, Samuel Kounev, Christian Becker; (2020).

[KWZ+20] A Survey on Predictive Maintenance for Industry 4.0. Christian Krupitzer, Tim Wagenhals, Marwin Züfle, Veronika Lesch, Dominik Schäfer, Amin Mozaffarin, Janick Edinger, Christian Becker, Samuel Kounev; (2020).

Chapter 1

Introduction

The introduction of this thesis motivates the research field addressed and introduces the three use cases from the domains of Intelligent Transportation Systems (ITS) and logistics as well as describes how Self-adaptive Systems (SASs) can be used in this context. It discusses shortcomings of existing approaches, and summarizes the problem statement. Then, it defines the research goals and research questions and presents the contributions of the thesis.

1.1 Motivation

In a world as dynamic as we find it today, where circumstances, processes, and requirements for systems—in this thesis a special focus is given to Cyber-Physical Systems (CPSs)[1]—are becoming increasingly complex, the challenges for these systems to be able to work in dynamic environments are also increasing. One of the most critical challenge for these systems is to analyze their environment and to adapt to changes accordingly. The SAS [CdLG$^+$09, KRV$^+$15] research area attempts to address this challenge. SASs can change their behavior and deal with changes in their environment and the system itself [Les20]. The trend towards SAS has given rise to several research communities, such as Organic Computing (OC) [MST17], Autonomic Computing (AC) [KC03], and Self-aware Computing (SeAC) [KLB$^+$17]. OC, for example, envisions to "enable future [Information and Communications Technology (ICT)] systems to carry out certain tasks on their own" [MST17, p.6] and thereby to "be able to adapt reasonably to changing requirements of their operating environment" [MST17, p.6]. This definition implies that OC systems are trying to shift

[1]We define CPSs as follows: CPSs are systems consisting of tightly integrated physical and cyber components interconnected through one or more networks. The cyber components consist of computing and communication facilities (local or remote, e.g., embedded systems or cloud services) used for monitoring, automating and controlling physical systems and processes. CPSs are normally based on complex feedback and control loops, where the physical components affect the cyber components and vice versa.

design time decisions to run time, putting the systems themselves in charge. Similar ideas drive the SAS and SeAC communities, as well as several further research directions. Most of these systems incorporate control loop mechanisms that allow one to react to changes in the environment or the system itself and adapt the behavior of the system accordingly. An example of a specific control loop concept from the SAS community is the Monitor Analyze Plan Execute Knowledge Control Loop (MAPE-K Control Loop) [KC03], while the OC and SeAC community introduced the Observer/Controller concept [TPB$^+$11] and the Learn-Reason-Act-Model Loop (LRA-M Loop) [KLB$^+$17], respectively. Since all of these concepts target similar goals, most of them can be transferred into each other [LKT19b].

In our daily lives, we are constantly in contact with SAS that aim to support and improve our way of life without us directly noticing it. Two example domains that directly influence our life are ITS and logistics that both are meaningful domains for the application of SAS and experience an increasing interest in the research community [GP19, JCS$^+$19, BGT$^+$19, MKPG19, SLHB21, JCS$^+$19, FVDW21, BBD$^+$21]. Hence, both domains provide complex and adaptable use cases to prototypical apply the contributions of this thesis. However, the contributions are not limited to these areas and can be generalized also to other domains such as the general area of CPSs and Internet of Things (IoT)[2] including smart grids or even intelligent computer networks. ITS encompasses all technological advances associated with sensors, communication and control of road traffic and transportation [Sus08], whereas we consider technological advances in logistics related to the processes for the efficient and effective transportation and storage of goods [oSCMP13]. For example, the first electric traffic signals in the domain of ITS is one use case for SASs that has led to the development of real-time traffic control in urban areas [XSBC14]. Another promising example use case in the domain of ITS is platooning, which targets addressing the challenge of rising infrastructure needs due to the constantly increasing traffic on roads. In Germany, for example, an increase in the total vehicle stock of about one million vehicles is observed from 2018 to 2019 [bus]. Authorities are trying to address the rising infrastructure demands by expanding the road network, but this leads to increased costs [FJM$^+$01] as well as critical issues for the environment and climate change. The demands on the infrastructure can be reduced through platooning, which is the ability of vehicles to travel with very close inter-vehicle distances, enabled by communication [RCC10].

[2]We define IoT as follows: The IoT consists of physical entities (things) that were not originally intended for communication with each other and with the environment. In IoT, these things are able to identify themselves, communicate, and interact via a network, based on Internet technologies. They can act depending on external triggers or local logic.

The use of platooning increases road throughput [Ala11], safety [RCC10] as well as reduces emissions and, hence, provides the possibility to reduce the environmental footprint [PD21]. While the feasibility of platooning has been demonstrated in diverse projects, the problem of platooning coordination still exists. Platooning coordination is the process of assigning vehicles to platoons and controlling the platooning activities. The platooning coordination problem is a multi-objective problem with several dimensions, such as objectives of the drivers, aspects of the platoon, and global traffic situation [SKSB21]. Further, fairness between participants must be ensured as the leading vehicle benefits less from slipstream effects [LKS$^+$21]. Platoons are usually coordinated using platooning coordination strategies implemented as part of a platooning coordination system. The latter is an example of SASs in the domain of ITS, as the coordination strategies can be considered as adaptation planning strategies that adapt the platoons, which together with the coordination system can be seen as comprising a CPS. We use platooning coordination as example for a prototypical case study to validate our contributions as it is a complex system, which consists of a set of autonomous entities that have individual objectives and requirements, and the application of SAS is particularly useful [LKT19b]. Still, the contributions of this thesis can also be applied to other use cases from the ITS domain for which the application of SAS appears meaningful. Further, the contributions can be generalized and applied also in other domains to which the application of SAS are meaningful such as in the management and control of smart grids.

Besides the prototypical application of SASs in the ITS domain, its concepts can be found in a variety of domains from which we focus on the logistics domain as it lately experiences an increased research interest [JCS$^+$19, FVDW21, BBD$^+$21]. This domain also provides complex use cases that can be enhanced using SAS concepts. Similarly, use cases from the logistics domain serve as a prototype to apply the proposed contributions of this thesis which can also be applied in other domains such as intelligent computer networks. An example in the logistics domain is the Vehicle Routing Problem (VRP), which deals with the planning of tours of road freight transport [LKK$^+$21a], while the Rich Vehicle Routing Problem (rVRP) addresses a variety of constraints to model a real-world problem. Over the past two decades, the demand for road freight transport has increased worldwide; in Germany, for example, it has increased by 150 billion ton kilometers to about 500 billion ton kilometers [Kor21]. Developments such as increased just-in-time production and online shopping (especially during the Covid-19 pandemic) will further push these numbers up in the coming years. To handle such a transport volume, efficient and correct route planning for transports is important. Therefore, fast and reliable solutions to the rVRP are required, which are difficult to compute as it is an NP-complete problem. A second example from the field of logistics is warehousing [LMK$^+$21a]. Warehouses play a central role in a company's supply chain and contribute to its logistical success. When humans are involved in warehousing, a distinction is made between picker-to-parts and parts-to-picker methods [DK07]. Experts estimate the picker-to-parts system to be the most widespread in Western Europe, accounting for over 80% [DKLDR07]. A well-known picker-to-parts system is the mezzanine warehouse. The work in a mezzanine warehouse consists of two main tasks: (i) filling the warehouse with goods (storage assignment) and (ii) picking items from the warehouse (order picking). Due to the NP-hardness and, thus, the complexity of the storage assignment and the order picking problem, efficient optimization algorithms are required to find adequate solutions in acceptable times. The two examples rVRP and mezzanine warehouses have in common that optimization algorithms are usually used to find solutions. In the context of SAS, these algorithms can also be considered as adaptation planning strategies. They optimize the tours, storage assignment, and order picking paths, and they usually have to handle changing demands, added orders, or changing environmental parameters. When using anytime optimization algorithms, such changes can be integrated at run time adapting the solutions accordingly. Therefore, optimization algorithms can be considered as adaptation planning strategies. Still, the contributions of this thesis can also be applied to other use cases from the logistics domain for which the application of SASs appears meaningful. Further, the contributions can be generalized and applied also in other domains to which the application of SASs are meaningful such as, for example, in the packet routing within intelligent computer networks.

1.2 Problem Statement and Shortcomings of Existing Approaches

In line with the No-Free-Lunch theorem [WM97][3], the proper selection of adaptation planning strategies is a key factor for the success of any SAS, as the performance of one strategy in a given context may not necessarily be transferable to other application scenarios. For example, one strategy might

[3]The No-Free-Lunch theorem states that "if an algorithm does particularly well on average for one class of problems then it must do worse on average over the remaining problems" [WM97, p.70]. This indicates that there is no single algorithm that performs best for all class of problems and, hence, a selection is required to match the algorithm to the problem class.

work well for platooning coordination in an urban scenario, while another strategy might work better in a highway scenario. In the year 1976, John R. Rice defined the algorithm selection problem, which involves finding the best performing algorithm for a given scenario [Ric76]. From then on, more and more research was done on algorithm selection, for example, in the field of machine learning, which describes this problem as a learning problem [SM09].

The idea of automatic algorithm selection has been transferred and used in a variety of applications, but little research has been done to generalize the approaches [SM09].

The observation from [FGKV19] that the choice of the strategy for adaptation planning in self-adaptive systems [CdLG+09, KRV+15] depends on the situation of the system opens up a wide set of applications where such a mechanism can be applied. Hence, a selection mechanism for adaptation planning strategies can be designed to be situation-aware, which turns out to be the main driver for the development of SASs [CMPPW20]. In the literature, clustering techniques are often applied to identify the current situation [FGKV19]. However, it is not enough to just recognize the situation. It is of utmost importance to use this knowledge and learn from it. This knowledge can be used to apply different strategies in different situations or to adjust the parameters of a strategy. Furthermore, it can be used in combination with previous experiences to learn in which situation which strategy and which parameter configuration work best. This combined learning and reasoning can be found in the SeAC research area, which has inspired the algorithms and mechanisms developed in this thesis. The literature provides diverse approaches to situation detection [CMPPW20, End17, LKPA15, RRFS07, HS19a, PRF16, KCP20], algorithm selection [SM09, KHNT19, KT19, KKHT15, BKK+16], and parameter optimization [NSW+12, FSH15, ZHO+18, CVV13, VCIC15].

However, no approach exists that combines situation detection, algorithm selection, parameter optimization into a learning and reasoning mechanism that is applicable to various use cases.

Up to this point, only systems as a whole have been considered. However, there are many systems that consist of several subsystems, which leads to the research area systems-of-systems or interwoven systems [BTW14]. For example the classical VRP specifies the assignment of customer orders to vehicles and the optimization of their tours [GRW08], commonly referred to as Traveling Salesman Problem (TSP). Hence, it can be classified as vertical or nested system-of-systems, which refers to a specific class of systems-of-systems. There are a variety of approaches that attempt to solve the VRP but do not consider two issues: (i) transferability of the solution to real-world applications under realistic assumptions and (ii) awareness of the system-of-systems structure of the problem. Regarding the first issue, Tim Pigden stated that the original model of the VRP does not match real-world applications since it does not include concepts of order, separate resources for the driver, tractor, and trailer [Pig13]. The rVRP extends this classical VRP with additional constraints required for a real-world application, such as Pickup and Delivery (P&D), Time Windows (TW), pause times, trailer capacities, and driver assignments. With respect to the second issue, considering the system-of-systems structure offers the advantage of designing approaches that take into account mutual effects of the nested VRP and TSP and adapt the choice of the appropriate algorithms to the current problem, situation, and objectives. Another example is the application of optimization algorithms to warehouse processes such as storage assignment and order picking. Numerous approaches to optimizing both of these warehouse problems can be found in the literature. Most approaches usually target one of them. Some works target both problems but view each problem separately and fail to integrate the interrelation between them [vGRCdK18]. However, [GGM10] has noted that both problems are strongly coupled. Therefore, optimizing each warehouse problem separately can lead to suboptimal solutions. Contrary to the VRP use case, the mezzanine warehouse use case can be considered as a representative of the horizontal system-of-systems class. An approach that takes into account this interrelatedness could provide various advantages, such as optimizing storage assignment regarding expected travel time for order picking.

> **The consideration of both classes of systems-of-systems is a critical factor for the performance of optimization approaches. These characteristics must be addressed at design time, but should also be taken into account when improving optimization techniques during lifetime.**

In this thesis, we make contributions addressing the above described challenges We address the problem of self-aware optimization of adaptation planning strategies by developing a framework that includes situation detection, strategy selection, and parameter optimization of the selected strategies. The framework applies concepts from SeAC and is able to learn from previous decisions. To address the systems-of-systems optimization problem, we propose two approaches for vertical and horizontal systems-of-systems. We apply the approaches to the platooning use case and the rVRP and mezzanine warehouse use cases, and we evaluate the positive impact of the contributions.

1.3 Research Questions

Based on the problem statement and shortcomings of existing approaches discussed above, we now define research goals and research questions. The first goal addresses the design of a self-aware optimization framework for adaptation planning strategies. The second goal focuses on improving the quality of optimization strategies when applied to complex systems-of-systems. In addition, we define a set of specific research questions per goal, which we address in the individual chapters. At the end of each chapter, we summarize the proposed approach and indicate which research questions were answered.

Goal A: *Self-aware optimization of adaptation planning strategies with particular attention to the field of ITS and logistics.*

This goal focuses on the design of a self-aware optimization framework and the implementation of a prototype concept, which we structure into six research questions. The first research question is concerned with identifying specific properties of the strategies under consideration and is answered in Chapter 6. The next four research question focus on the design of the framework as well as the consideration of the specific properties of strategies identified in the first research question, addressed in Chapter 7. The last research question focuses on the performance evaluation of the framework presented in Chapter 10.

RQ A.1: What are the specific characteristics of adaptation planning strategies in ITS and logistics?

RQ A.2: How can we design a self-aware optimization framework for adaptation planning strategies that reduces the manual effort for algorithm selection and parameter optimization?

RQ A.3: How can the framework support situation-awareness?

RQ A.4: How to design adaptation planning strategies to support fairness in systems with autonomous entities focusing on the platooning use case?

RQ A.5: How to address uncertainty in adaptation planning strategies to cope with dynamics of the environment focusing a route planning use case?

RQ A.6: How does the proposed framework compare to state-of-the-art adaptation planning strategies from the ITS domain?

Goal B: *Improving the quality of optimization strategies in complex systems-of-systems with a special attention to the field of logistics.*

The second goal of this thesis focuses on the optimization of complex systems-of-systems in the field of logistics. It is divided into two main research questions, the first of which addresses the class of vertical system-of-systems applied on the rVRP use case and the second of which addresses the class of horizontal system-of-systems applied on the mezzanine warehouse use case. The first and second subordinate research questions of the first main research question cover the discussion and integration of constraints and objectives in flexible optimization workflows for nested systems and are answered in Chapter 6 and Chapter 8. The third subordinate research question addresses the evaluation of the proposed approach and is answered in Chapter 11. The first and second subordinate research questions of the second main research question address the constraints and objectives as well as the design of optimization procedures in coexisting systems and are answered in Chapter 6 and Chapter 9. The third subordinate research question concerns the evaluation of the proposed approach and is answered in Chapter 12.

RQ B.1: How to optimize nested systems-of-systems considering their interdependence using rVRP as an example?

- **RQ B1.1:** How can real-world constraints and objectives be integrated into an optimization approach using the example of the rVRP?
- **RQ B1.2:** How can a workflow be designed that enables flexible optimization of nested systems?
- **RQ B1.3:** To what extent do the proposed optimization approaches outperform the state-of-the-art in the example use cases VRP and TSP?

RQ B.2: How can coexisting systems-of-systems be optimized considering the influence of their interaction using mezzanine warehouses as an example?

- **RQ B2.1:** How can domain constraints be considered in the optimization of real-world problems consisting of coexisting systems-of-systems taking the example of mezzanine warehouses into account?
- **RQ B2.2:** How can multi-objective optimization approaches be designed that aim at good overall system performance rather than optimizing a single system?
- **RQ B2.3:** Considering existing optimization algorithms for the independent systems, to what extent do the proposed approaches improve the solution quality for horizontal systems-of-systems optimization?

1.4 Contributions

This section presents the core contributions of this thesis and maps them to the research questions in Section 1.3. The first contribution focuses on the analysis of adaptation planning strategies in the targeted domains.The following three contributions deal with the design of a self-aware optimization framework and how to consider the specific characteristics of adaptation planning strategies as part of it. The last two contributions aim at optimizing problems with a system-of-system structure including both vertical and horizontal systems-of-systems.

Contribution 1: **Analysis of Specific Characteristics of Adaptation Planning Strategies**

> This contribution addresses the research question **RQ A.1** of **Goal A** and is part of our papers [LNH+21, LHKK21b, LKS+21, LKK+21a, LMK+21a]. By conducting a set of case studies in the ITS and logistics domain and analyzing which specific characteristics are present. For this purpose, we select platooning and route planning problems in highly dynamic environments from the ITS domain and the rVRP and mezzanine warehouse optimization problem from the logistics domain. Using these case studies, we derive the need for situation-aware optimization of adaptation planning strategies and argue that fairness is an important consideration when applying adaptation planning strategies in ITS. We also observe that it is important to consider uncertainties in the environment when planning adaptations. Furthermore, we analyze the system-of-systems structure of the case studies rVRP and mezzanine warehouses and identify the nested and coexisting system-of-systems structure in these studies.

Contribution 2: **Component-based Framework for Self-Aware Optimization of Adaptation Planning Strategies**

> This contribution focuses on the research questions **RQ A.2**, **RQ A.3**, and **RQ A.6** of **Goal A**. This contribution is addressed in our papers [Les20, LHKK21a, LHKK21c]. Here, we propose a novel self-aware optimization framework for adaptation planning strategies that is designed to be generically applicable. That is, the framework can be applied to a wide range of use cases because it abstracts adaptation planning strategy selection and parameter optimization into a separate layer placed above the application and the adaptation planning system. We propose a three-layer system model in which the framework operates in the top layer. The framework

offers use case independent interfaces defined by a Domain Data Model (DDM), which an operator can use to apply the framework by defining available adaptation planning strategies, parameters to be optimized, and performance measures. These inputs are then used by the framework to automatically select and optimize adaptation planning strategies. The framework is composed of four main components: (i) Coordination, (ii) Situation Detection, (iii) Strategy Selection, and (iv) Parameter Optimization. While the coordination component receives observations from the application and triggers the other components, the situation detection component is responsible for applying a rule base or clustering techniques to identify the current application situation. The strategy selection uses the identified situation and combines it with learned best practices from previous decisions, selecting an adaptation planning strategy that is expected to work best in the current situation. Finally, the parameter optimization component receives the current and all historic observations of the identified situation and selected strategy, and it optimizes the parameters of the latter. We evaluate all components and the overall performance of the framework on the platooning coordination use case and compare it with state-of-the-art mechanisms.

Contribution 3: **Fairness-Ensuring Adaptation Planning Strategies**

This contribution focuses on research question **RQ A.4** of **Goal A** and is addressed in our paper [LKS+21]. The self-aware optimization framework proposed in the previous contribution encompasses the situation-dependent behavior of adaptation planning strategies and, hence, this aspect does not need to be addressed separately. However, we propose fairness-ensuring adaptation planning strategies to address fairness issues at the application level using platooning as example scenario. To this end, we analyze the uniform distribution of positive and negative effects for platooning participants and design six different vehicle sequence rotation strategies for platoons that compensate for these effects.

Contribution 4: **Addressing Uncertainty in Adaptation Planning Strategies**

This contribution addresses research question **RQ A.5** of **Goal A** and is part of our paper [LHKK21b]. Similar to the previous contribution, we propose to address uncertainty explicitly as part of the adaptation planning strategies using route planning as example scenario. We analyze the existence of uncertainties in a highly dynamic use case by planning refueling along a route. We propose a methodology to account for this uncertainty and incorporate uncertainty measures in the utility functions used for fueling planning. Further, we design six utility functions that consider different aspects of route planning such as the distance, expected costs, and duration. For two of these functions, we incorporate uncertainty measures for dynamic fuel prices by adding penalties for longer travel time or longer distance to the next gas station.

Contribution 5: **Optimization of Nested Systems-of-Systems**

This contribution addresses research question **RQ B.1** and its subordinate research questions of **Goal B**. This contribution is addressed in our publications [LKK$^+$21a, LKK$^+$21b]. We explore the optimization of nested systems-of-systems. We first define our use case rVRP and analyze the constraints and objectives that must be considered when formulating a real-world rVRP and how they can be integrated into a solution approach. Then, we analyze the nested structure of rVRP and design an integrated workflow that allows for individual but also flexible and interchangeable optimization of both systems. The workflow consists of two stages and addresses specific constraints such as time windows and pause times. We apply a Genetic Algorithm (GA) and an Ant Colony Optimization (ACO) algorithm to both the VRP and TSP stages and compare their performance with state-of-the-art algorithms for this use case.

Contribution 6: **Optimization of Coexisting Systems-of-Systems**

The last contribution addresses the research question **RQ B.2** and its subordinate research questions of **Goal B**. This contribution is addressed in our publications [LMK$^+$21a, LMK$^+$21b]. Here, we study coexisting systems-of-systems and define our use case in mezzanine warehouses. In this type of warehouses, the processes of storage assignment and order picking form the two main tasks. We first analyze which ergonomic and economic constraints and objectives apply for a real-world optimization problem and how they can be integrated into an optimization approach. Then, we design an integrated approach that simultaneously optimizes several ergonomic and economic constraints and aims for a good overall system performance, taking into account the interdependence of both systems. For this purpose, we use Non-dominated Sorting Genetic Algorithm II (NSGA-II) for storage assignment and ACO for order picking, and adapting them to the specific requirements of the horizontal system-of-systems structure. We evaluate the proposed approach against state-of-the-art approaches for these processes and show the positive impacts of our integrated approach.

The presented contributions represent an important advance for both the academic research community and practical applications leading to a major advancement in the field of SASs and systems-of-systems applied on CPSs. To the best of our knowledge, we are the first to design a self-aware optimization framework for adaptation planning strategies integrating situation-awareness, algorithm selection, parameter tuning, as well as learning and reasoning. The promising results of the framework, which performs close to the gold standard in our evaluation, demonstrate the importance of this contribution and offer new opportunities to address the underlying problems from a software engineering perspective. In addition, our proposed adaptation planning strategies to compensate negative effects of platooning provide an important contribution and starting point for further research in platooning coordination, which could lead to higher acceptance in society and more likely adoption of the technology in the real world. Our mechanism and utility functions for coping with uncertainty are an important step to improving the capabilities of SASs in an increasingly turbulent environment. Moreover, our contributions to nested and coexisting systems-of-systems optimization in logistics provide a major contribution to research and practical applications.

Finally, we select real-world use cases for our approaches and cooperated with industrial partners, highlighting the practical relevance of our contributions. Considering real-world use cases and reducing expert knowledge and manual effort in selecting the most appropriate strategy and its parameters is a major step in bridging the gap between academia and practice. For example, our approach on the rVRP is already integrated into our partner's software and improves both time to solution and solution quality. In conclusion, the contributions of this thesis have spawned several research projects such as a long-term industrial project on optimizing parcel deliveries funded by *Bayerisches Verbundforschungsprogramm* (*BayVFP*) *– Digitalisierung* and further collaborations, opening up many promising avenues for future research.

1.5 Outline

The remainder of this thesis is divided into three parts. Part I introduces the background and foundations necessary to understand this work and is composed of five chapters. Chapter 2 presents the concept of Self-aware Computing (SeAC), Chapter 3 introduces the field of optimization, Chapter 4 presents platooning and its coordination as part of ITS, and Chapter 5 introduces foundations of the addressed VRP and mezzanine warehouse problems. Chapter 6 presents selected motivating scenarios and analyzes specific properties of the addressed use cases. Part II contains three chapters and presents the main approaches of this work. Chapter 7 presents the concept of a self-aware optimization framework and addresses situation-awareness, fairness, and uncertainty. Chapter 8 proposes a workflow to address vertical systems-of-systems while Chapter 9 introduces the integrated approach on optimizing horizontal systems-of-systems. Part III presents the evaluation of the proposed approaches and consists of three chapters. Chapter 10 evaluates the proposed self-aware optimization framework on the platooning use case. Chapter 11 analyzes the workflow on vertical systems-of-systems while Chapter 12 discusses the approach on horizontal systems-of-systems. Finally, Part IV concludes this work and is composed of three chapters. Chapter 13 presents related work, Chapter 14 summarizes the thesis, and Chapter 15 outlines future work.

Part I

Foundations

Chapter 2

Self-aware Computing

The SAS [CdLG+09, KRV+15] research area seeks to address challenges that arise from dynamic environments and changing demands on the system. The SAS can change their behavior and deal with changes in their environment and the system itself [Les20]. The trend toward SAS has given rise to several research communities, such as OC [MST17], AC [KC03], as well as SeAC [KLB+17]. According to [KLB+17, p. 5] SeAC can be defined as follows.

"Self-aware computing systems are computing systems that:

> 1. *learn models* capturing *knowledge* about themselves and their environment (such as their structure, design, state, possible actions, and runtime behavior)
> 2. *reason* using the models (e.g., predict, analyze, consider, and plan) enabling them to *act* based on their knowledge and reasoning (e.g., explore, explain, report, suggest, self-adapt, or impact their environment)
>
> in accordance with *higher-level goals*, which may also be subject to change."

These systems observe their environment and attempt to learn models about themselves and the characteristics of the environment. Then, SeAC systems use these models to reason about their current state and their interaction with their surroundings to plan adaptation actions aimed at better performance in the specific environment. Thus, the two major distinctive characteristics of a SeAC system are learning and reasoning. These processes are executed continuously, enabling the system to operate in dynamic environments and adapt to a changing environment without the need for human interaction.

Most of the above mentioned research areas proposed mechanisms or control loops to react to changes in the environment or the system itself and adapt the behavior of the system. A representative of the SAS community is the MAPE-K Control Loop [KC03], while the OC community introduced the Observer/Controller concept [TPB+11]. The SeAC domain defined the LRA-M

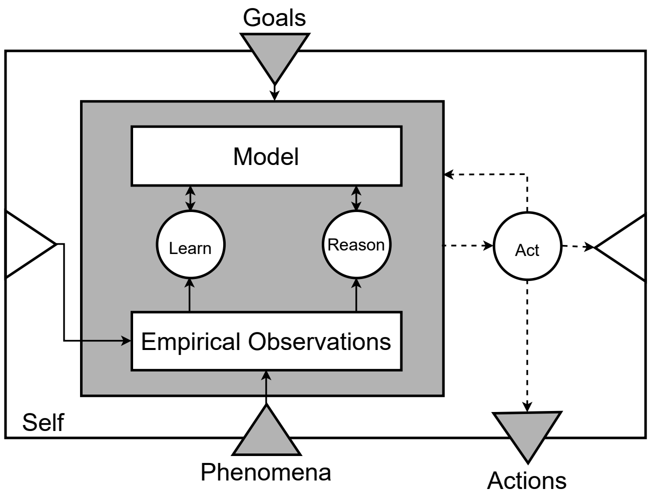


Figure 2.1: Concept of the LRA-M Loop as used by the SeAC research community [KLB+17]. The main components of this loop are the Empirical Observations, the learned Models as well as the ongoing Learn and Reason processes.

Loop [KLB+17]. Since all of these concepts aim at the same vision, most of them can be transferred into each other [LKT19b]. Figure 2.1 represents the LRA-M Loop loop as used by the SeAC research community. The system referred to by the word *Self* receives higher level goals that guide the direction of adaptation decisions. The self perceives phenomena such as characteristics of the environment, as well as information about itself, and stores them in the component *Empirical Observations*. This component is updated with each new observation and serves as data base for the ongoing *Learn* and *Reason* processes. The *Learn* process attempts to abstract perceived behavior of the environment, as well as effects of one's decision on the environment, into models and updates them as new phenomena are sensed. The *Reason* process also bases on the *Empirical Observations* and aims to analyze the sensed phenomena to explore and explain its environment. It also plans adaptations of the system, which are passed to the *Act* component. This component executes the given *Actions* to fulfill planned adaptations.

Chapter 3

Optimization

"Optimality is a fundamental principle, establishing natural lows, ruling biologic behaviors, and conducting social activities. Therefore, optimization started from the earliest stages of human civilization." [DPW09, p.1539]

Optimization has emerged into an important interdisciplinary research area including mathematics, computer science, industrial engineering, and management science [DPW09]. Optimization can be defined as "the act of obtaining the best result under given circumstances" [Rao19] with the goal of minimizing the required effort or maximizing the desired gain. Years of research in this area have led to various contributions, from the definition of optimization problems to the design and development of diverse algorithms and the evaluation of found solutions. The purpose of this chapter is to provide a brief introduction to the topic of optimization and to point the reader to literature that can be used for further familiarization. It is based on our publication [LMK+21b]. Therefore, Section 3.1 gives an overview of the formal definition of optimization problems, objectives and constraints as well as a classification of these problems. Furthermore, Section 3.2 summarizes the concepts of multi-objective optimization and Pareto dominance. Then, Section 3.3 presents quality indicators for assessing solutions of multi-objective optimization problems. Finally, Section 3.4 categorizes optimization algorithms and presents the approaches used in this thesis.

3.1 Optimization Problems

Mathematically, an optimization problem can be defined as [Rao19]:

$$\text{Find } X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ which minimizes } f(X) \tag{3.1}$$

subject to the constraints

$$g_j(X) \leq 0, \qquad j = 1, 2, \ldots, m \tag{3.2}$$

$$l_j(X) = 0, \qquad j = 1, 2, \ldots, p \tag{3.3}$$

where X is the design vector with n dimensions comprising the variables to be determined by the optimization process. $f(X)$ is the objective function and $g_j(X)$ and $l_j(X)$ are inequality and equality constraints, respectively. Constraints limit the range of values to which the design variables can be set, and, thus, represent functional and other requirements on the solution. Constraints can be classified into two types: (i) hard constraints and (ii) soft constraints. While hard constraints must be satisfied to find a feasible solution, the soft constraints should be satisfied, and any failure to satisfy with this constraint is penalized in the objective function. A feasible solution is called locally optimal if the neighborhood of this solutions does not contain solutions with better objective function values. A feasible solution is called globally optimal if all possible solutions in the design space achieve lower objective function values. The goal of optimization processes and optimization algorithms is to find the globally optimal solution. When the complexity of an optimization problem does not allow finding the globally optimal solution, an optimization algorithm returns the best solution found so far. According to [Rao19], optimization problems can be classified into several categories: based on the existence of constraints, based on the nature of design variables, based on the physical structure of the problem, based on the nature of equations involved, based on the permissible values of the design variables, based on the deterministic nature of variables, based on the separability of the functions, and based on the number of objective functions.

3.2 Multi-objective Optimization

When considering optimization problems, multiple and even conflicting objectives must often be considered [NZES05]. Traditionally, these objectives are integrated into a single objective function by aggregating all objectives with predefined weights or converting them into constraints. According to [NZES05], this leads to four limitations. First, defining the aggregated objective function requires a priori knowledge about the importance of each objective. Second, aggregating the objectives into a single function leads to a single solution. Third, this leads to the impossibility to balance the objectives in a set of solutions. Fourth, it is possible that a solution cannot be obtained unless the search space is convex. Therefore, this process of objective reduction is not feasible for complex multi-objective optimization problems, leading to the concept of Pareto fronts. Pareto optimality theory aims to balance a set of possibly conflicting objectives to find a number of solutions that perform equally well [WAY+16, p.632]:

> "Pareto optimality defines dominance to compare solutions, i.e., a solution A is said to dominate another solution B, if for all objectives, A is no worse than B at the same time at least one objective exists that A is better than B."

The resulting set of solutions that cannot be dominated by other solutions is called Pareto front. The set of all non-dominated solutions that exist in a design space are called optimal Pareto front, while the Pareto front determined by an optimization algorithm is called computed Pareto Front [WAY+16, NZES05].

3.3 Quality Indicators for Multi-objective Optimization

To assess the quality of a Pareto front, we summarize the following performance indicators as introduced by [WAY+16]. These metrics use the concept of a reference Pareto front (PF_{ref}) or an optimal Pareto front. The performance indicators can be categorized into four categories: (i) the Coverage aspect evaluated using Coverage (C), (ii) Convergence evaluated using Generational Distance (GD) and Euclidean Distance (ED), (iii) Diversity evaluated using Pareto Front Size (PFS) and Generated Spread (GS), and (iv) Combination evaluated by Inverted Generational Distance (IGD) and Hypervolume (HV).

First, the C performance indicator defined in Equation (3.4) quantifies the extent to which a computed Pareto front (PF_c) covers the reference Pareto front, i.e., the number of solutions (s) of the computed Pareto front that are also part of the reference Pareto front divided by the number of solutions in the reference Pareto front. The best value for this metric is one since this indicates that the computed Pareto front covers the whole reference Pareto front which represents the best known solutions.

$$C = \frac{|\cup_{s \in PF_c} s \in PF_{ref}|}{|PF_{ref}|} \tag{3.4}$$

Second, the GD performance indicator measures the Euclidean distance from each solution in the computed Pareto front to the nearest solution in the reference Pareto front ($d(s_i, PF_{ref})$) and is defined in Equation (3.5). Hence, it provides a measure to judge how much distance is present between the computed and the best known solution. The best possible result for this metric is zero, as it indicates that the computed Pareto front directly covers the whole reference Pareto front without any distances.

$$GD = \frac{\sqrt{\sum_{i=1}^{|PF_c|} d(s_i, PF_{ref})^2}}{|PF_c|} \tag{3.5}$$

Third, the ED performance indicator defined in Equation (3.6) measures the Euclidean distance from a reference solution to its closest solution in the computed Pareto front ($d(s_{ref}, PF_c)$). The reference solution is defined similar to the reference Pareto front and selects the best known values among all retrieved solutions for each objective and combines them into one solution value. The best possible value for this metric is zero, indicating that the best solution of the computed Pareto front matches the reference solution.

$$ED = d(s_{ref}, PF_c) \tag{3.6}$$

Fourth, the PFS performance indicator measures the number of solutions in the computed Pareto front as defined in Equation (3.7). A larger PFS value indicates a higher diversity of the computed solutions and, hence, the user has more options to choose from.

$$PFS(PF_c) = |PF_c| \tag{3.7}$$

Fifth, the GS performance indicator defined in Equation (3.8) measures the diversity of the solutions that exist in the computed Pareto front. Therefore, we calculate a set of extreme solutions from the reference Pareto front called $e_1, \ldots, e_m$ where e_i has the best value for the i-th objective function (f_i). $d(e_k, PF_c)$ refers to the Euclidean distance from extreme solution e_k to its nearest solution in PF_c. $d(s, PF_c)$ calculates the Euclidean distance from the solution $s \in PF_c$ to its nearest solution in PF_c and $\bar{d}$ represents the mean of these distances over all solutions in PF_c. This performance indicator measures how even the solutions in PF_c are spread, where a lower GS value indicates a more even distribution.

$$GS(PF_c, PF_{ref}) = \frac{\sum_{k=1}^{m} d(e_k, PF_c) + \sum_{s \in PF_c} \left| d(s, PF_c) - \bar{d} \right|}{\sum_{k=1}^{m} d(e_k, PF_c) + |PF_c| \cdot \bar{d}} \tag{3.8}$$

Sixth, the IGD performance indicator combines convergence and diversity aspects of PF_{ref} and is defined in Equation (3.9). Again, we use the Euclidean distance of a solution s_i in PF_{ref} to its closest solution in PF_c which we define as $d(s_i, PF_c)$). We set the sum over all solutions in the reference Pareto front in the ratio to the size of the reference Pareto front. We use this value to compare two computed Pareto fronts where the computed Pareto front with the lowerIGD value is closer to the reference Pareto front, and hence, we consider it better than the other computed Pareto front.

$$IGD(PF_c, PF_{ref}) = \frac{\sqrt{\sum_{i=1}^{|PF_{ref}|} d(s_i, PF_c)^2}}{|PF_{ref}|} \tag{3.9}$$

Finally, the HV performance indicator defined in Equation (3.10) also deals with the combination of convergence and diversity aspects of the computed solutions. It measures the volume in the objective space that the PF_c covers with respect to a given reference point. Thus, it computes the volume of the hypercube resulting from the diagonal corners v_i between the solutions s_i in the PF_c and the reference point P_{ref} having the worst objective function values. The reference point must not overlap with the values of PF_c, and can therefore be defined outside the range of values of the objective functions—below the value range of values for a maximization problem and above the range for a minimization problem. A higher HV value in general indicates a better performance of the PF_c.

$$HV(PF_c, P_{ref}) = volume(\cup_{i=1}^{PF_c} v_i) \tag{3.10}$$

3.4 Optimization Algorithms

Research in the field of optimization algorithms is a very old discipline and, accordingly, has already developed a wide range of techniques. Following the structure of [Rao19] and [Bro11], optimization algorithms can be divided into the following categories: exact algorithms, mathematical programming techniques, stochastic algorithms, physical algorithms, probabilistic algorithms, evolutionary algorithms, swarm algorithms, immune algorithms and neural algorithms. For a comprehensive overview of common algorithms from these categories, we refer the reader to [Rao19, Bro11]. In the following, we limit the discussion of optimization algorithms to those used in this thesis.

The first category of optimization algorithms are exact algorithms with the most general brute-force algorithm [PK87]. This algorithm does not require any domain-specific knowledge, but operates on a set of states, starting from an initial state and using legal operators. Breadth-first search or depth-first search are two specific examples of brute-force algorithms. The brute-force algorithms perform an exhaustive search and explore the entire design space, which is the reason for their exponential time complexity. However, the advantage of these algorithms is that they are guaranteed to find the best solution.

Local Search (LS) is an example algorithm from the category of stochastic algorithms [Bro11, LMS01]. This algorithm starts from an initial constructed solution and explores the direct neighborhood of that solution. Usually, the neighborhood can be found by replacing one decision variable of the current solution. If the neighboring solution has better objective function values, the algorithm uses it as the next starting point. If the neighbor solutions do not lead to better values, LS terminates and returns the solution. If the LS examines all neighboring solutions and selects the one that maximizes the objective function value, this variant is called Hill Climbing.

From the category of probabilistic algorithms, we consider Bayesian Optimization [PGCP00, Bro11]. This algorithm builds a probabilistic model of the joint distribution of promising solutions using Bayesian networks. At each iteration, the algorithm queries one observation point from the objective function and updates its model accordingly. An additional acquisition function defines the most promising candidate for the next observation. The algorithm terminates after a number of iterations and returns the current best solution.

Simulated Annealing is one of the physical optimization algorithms [KGV83, Bro11]. It was inspired by the physical process of annealing in metallurgy and involves heating and cooling metal to increase the strength and durability of the material. The idea is that as the temperature of a material increases, the degrees of freedom within the system also increase and more changes are possible. As the temperature decreases, the possibility of changes to the system becomes smaller. The algorithm starts with an initial solution and changes it iteratively. Therefore, two functions are relevant: a temperature function and the objective function. As long as the temperature is still high, the objective function has less influence on the selection of the next candidate solution. With decreasing temperature, the objective function gets more weight and the algorithm only considers solutions with better objective function values.

From the category of evolutionary algorithms, we consider the GA [Hol92, Bro11]. It is inspired by the process of natural selection and the evolution of a population by recombination and mutation of individuals representing a solution encoded as a genome. The algorithm starts with an initial set of solutions, called the population, and performs selection, crossover, and mutation in each iteration. Selection uses the objective function to evaluate the fitness of individuals and select the individuals for recombination. Then, the crossover procedure combines the genomes of the selected individuals to breed a new individual as part of the offspring. Each new individual is then mutated to increase the diversity of the population. Finally, as the population increases in each iteration, the individuals with the lowest fitness value are discarded in each iteration to achieve the predefined population size. A multi-objective version of the GA is the NSGA-II, which computes a Pareto front and aims at a good distribution of solutions by considering the crowding distance as density estimate of the solutions in the front [DPAM02].

Finally, ACO is a representative of swarm algorithms [DMC96, Bro11]. ACO is inspired by the behavior of ants in search of food and uses the concept of pheromones that ants leave on a good path between their colony and food sources. The algorithm first simulates a random movement of ants in the environment. Once an ant discovers a good food source, it begins emitting pheromones along the path back to the colony. When other ants notice the pheromone trail, they follow it and increase the concentration of pheromones along the way. However, some of the ants do not follow the path and keep exploring new paths to find better routes and create new pheromone trails. As the pheromone in the environment decreases, the shortest path is receives the highest pheromone concentration during the optimization process. After a predefined period of time, the algorithm terminates with the shortest path represented by the pheromone trail with the highest concentration.

Chapter 4

Intelligent Transportation Systems

The concept of ITS arose in the early 20th century with the first electric traffic signals being considered as the original ITS in 1928 [FJM$^+$01]. Since then, the interest has spread to the whole world and the main contributions have been produced by Europe, U.S., and Japan [FJM$^+$01]. In the 1970s, the U.S. introduced the Electronic Route Guidance System (EGRS) as initial stage of ITS [ALS11]. In Europe, the countries Germany, United Kingdom, and France began to thoroughly study the ITS technology in the 1980s, founded the European Road Transport Telemetric Implementation Coordination Organization (EUREKA) in 1985, and started the PROMETHEUS project in 1986 [ALS11]. In the year 1986, an informal group of academics, federal and state transportation officials as well as people from the private sector started a discussion on the road infrastructure and current challenges in traffic, which lead to the group "Mobility 2000" that produced a landmark document in 1990 [Sus08]. Also Japan started ITS development in the 1980s such as the Japanese projects Road Automobile Communication System (RACS) and Advanced Mobile Traffic Information and Communication System (AMTICS) [FJM$^+$01].

Diverse definitions of ITSs can be found in the literature. Sussman provides a very broad definition in his article of 2008 *Perspectives on Intelligent Transportation Systems* [Sus08, p.3]:

> "ITS combines a high technology and improvements in information systems, communication, sensors, and advanced mathematical methods with the conventional world of surface transportation infrastructure."

The EU defines ITS in the directive 2010/40/EU [EU221]:

> "Intelligent Transport Systems or ITS means systems in which information and communication technologies are applied in the field of road transport, including infrastructure, vehicles and users, and in traffic management and mobility management, as well as for interfaces with other modes of transport."

While the first definition is very broad and names some involved technologies, the second definition offers more application scenarios and appears more specific. In this work, we consider both definitions as relevant and fitting for the use case scenarios we discuss.

In addition to the definition of ITS, Sussmann also identified the need for the "ITS-4" technologies [Sus08]: (i) the ability to sense, (ii) the ability to communicate, (iii) the ability to process, and (iv) the ability to use this information properly and in real-time. He states that this technological innovation is required for the development of future ITS. Further, he defines six areas of application for ITS: (i) Advanced Traffic Management Systems (ATMS), (ii) Advanced Traveler Information Systems (ATIS), (iii) Advanced Vehicle Control Systems (AVCS), (iv) Commercial Vehicle Operations (CVO), (v) Advanced Public Transportation Systems (APTS), and (vi) Advanced Rural Transportation Systems (ARTS). For this work, the AVCS are especially important as they can be regarded as a further development of the driver's control of a vehicle such as collision detection and warning [MKES21]. Further, automatic breaking systems are relevant in this area. Additionally, Sussman mentions the term platooning, where cars drive in groups of ten or more vehicles with close inter-vehicle gaps at normal highway speed under automatic control. The development of autonomous vehicles in the last decades enables the development and deployment of platooning in the real world. While SAS can be applied to all the mentioned areas of ITS and could further develop the respective systems, in this work we use platooning as example for a prototypical case study. Platooning provides a meaningful opportunity to validate the contributions of this thesis as it is a complex system, which consists of a set of autonomous entities that have individual objectives and requirements, and the application of SAS is particularly useful [LKT19b]. Still, the contributions of this thesis can also be applied to other use cases from the ITS domain for which the application of SAS appears meaningful such as traffic management using traffic lights [PRT$^+$08] or traffic cameras [VWMA11].We refer the interested reader to a survey by Malik et al. [MKES21] in which solution approaches and future research directions on collaborative autonomous driving are discussed.

In the following, Section 4.1 introduces the concept of platooning including the different levels of platooning as well as platooning control. Then, Section 4.2 presents platooning coordination applies a taxonomy and classifies existing platooning coordination approaches. Finally, Section 4.3 introduces simulators that can be used for platooning and its coordination.

4.1 Platooning

Similar to the research in the general ITS domain, the research on platooning became more and more important in the literature. Platooning describes a cooperative driving technologies of automated vehicles to travel with small inter-vehicle distances of 5–10 meters [RCC10]. Those vehicles benefit from slipstream effects due to air drag reduction resulting in energy savings [Seg16]. However, the lead vehicle experiences reduced fuel savings and, in some platooning approaches, its driver has to drive manually, whereas other vehicles can follow in a self-driving mode; hence, those drivers do not have to control their vehicles. Besides these advantages for individual vehicles, further advantages for the overall traffic can be observed such as increased traffic flow, increased capacity of the existing road infrastructure, safety, and comfort benefits [TJS16, RCC10]. According to Sturm et. al [SKSB21], the objectives of platooning participants are spread among individual vehicles, the platoon, and the global system. They assign the following objectives to the individual vehicles: energy efficiency, vehicle safety, velocity, time, user comfort, destination and distance, balance of individual objectives, and cost balancing. The objectives of the platoon consist of energy efficiency, platoon safety, and the balance of platoon objectives. Finally, the global system aims at energy efficiency, global safety, traffic flow and road capacity, and a balance of global objectives. Consequently, the coordination of platooning, that is, the assignment of vehicles to platoons, is a challenging task as it represents a multi-level, multi-objective optimization problem [KSB$^+$18].

The research on platooning already started in the early 1970s and since the 1980s, several projects described platooning concepts. The research focus of these platooning projects shifted over time from enabling autonomous driving capabilities and communication-supported cooperative driving behavior towards platooning coordination and, more recently, multi-brand platooning and real-live demonstrations. We refer the interested reader to our survey on platooning coordination, where we summarize all relevant platooning coordination projects [LBS$^+$21].

4.1.1 Levels of Platooning

In this work, we distinguish two levels of platooning similarly to our definition in [LBS$^+$21]: (i) platooning control, and (ii) platooning coordination. We define both terms, delineate them by explaining our understanding of both levels, and clarify the level on which this work focuses.

Platooning Control is the control of a single vehicle on the lowest possible level including maintaining the distance, sending braking signals or signaling platoon members to overtake another vehicle.

Platooning Coordination includes the management of (i) the composition of a platoon, (ii) inter-platoon interactions as well as (iii) interactions between other vehicles and platoons.

Hence, we refer to all actions performed by a Cooperative Adaptive Cruise Control (CACC) controller, including longitudinal control, lateral control, and string stability, as platooning control. In contrast, platooning coordination operates at a higher level and coordinates the composition of platoons as well as intra-platoon and extra-platoon maneuvers. Hence, coordination regarding platooning is possible on two levels: (1) between platoons and other platoons or vehicles as well as (2) within a platoon. For both types, we assume the presence of a platooning control approach to maintain the distances between vehicles in a platoon at any time. Platooning coordination typically incorporates for instance (i) finding a suitable platoon for a vehicle, (ii) managing inter-platoon interactions, such as merging platoons or (iii) routing platoons.

4.1.2 Platooning Control

Realizing platooning in practice requires a set of technologies, including control and communication systems. Control systems for platooning have two components: longitudinal control—accelerating and braking the vehicle to maintain a target distance to the front vehicle— and lateral control—steering the vehicle. Further, communication is a fundamental component of platooning as it is (i) useful for lateral control, (ii) essential for longitudinal control, and (iii) responsible for coordinating platooning activities. This section is taken from our paper [LBS+21], in which we provide a brief overview of some concepts required for realizing platooning on the lowest possible level, i.e., the control of single vehicles within a platoon. We do not review the literature in all its depth, as the focus of this work is on the higher levels. Still, this section can be useful to the reader to grab a general understanding of the application. In addition, we mention relevant literature at appropriate passages for the interested reader in order to enable an in-depth acquaintance with this topic.

A fundamental part is the longitudinal control, which is realized through a control system computing acceleration commands to maintain a desired inter-vehicle gap. This control system is named CACC, which derives from the standard Adaptive Cruise Control (ACC) [Raj12]. The term *cooperative* indicates that vehicles, together with data coming from radars, lidars, and cameras [UAB+08, LAB+11], exchange state information used to compute the control action by means of communication. In contrast, ACC only takes locally sensed data into account to make decisions. The introduction of communication brings several benefits, including reduced inter-vehicle spacing and faster reaction to changes in dynamics [Raj12]. There is a vast literature of approaches targeting the design of a CACC system, which differ by control technique and assumptions on input data.

Regardless of the technique, all control algorithms must ensure a fundamental property called *string stability*, meaning that errors occurring at the head of the platoon must not be amplified but be dampened towards the tail. More formally, let δ_i be the spacing error (i.e., the difference between the target distance and the actual one) between vehicle i and its predecessor and let $H(s) = \frac{\delta_i}{\delta_{i-1}}$ be the transfer function relating spacing errors between consecutive vehicles. A CACC is said to be string-stable [Raj12] if the following conditions hold:

$$||H(s)||_\infty \leq 1 \qquad h(t) > 0, \forall t \geq 0. \tag{4.1}$$

The left condition ensures that the magnitude of the errors is attenuated towards the tail, while the second ($h(t)$ is the impulse response of $H(s)$) ensures that the errors must have the same sign. It is not sufficient to dampen the magnitude towards the tail, but we must also avoid a vehicle being too close to its predecessor (negative error) and its follower being too far (positive error) and vice versa. This is one of the possible definitions of string-stability but it might need to be adapted to the control system being proposed [FZL+19, PSvdWN14, XPN16]. As we will briefly describe, different CACC have different string-stability properties depending to the inputs they consider.

As an example, in the 2000s the PATH project [RTLZ00] defined a CACC which is still commonly considered by researchers in the field. The control formula for the vehicle in position i is defined as

$$\begin{aligned} u_i &= (1 - C_1)a_{i-1} + C_1 a_0 \\ &- \left(\left(2\xi - C_1\left(\xi + \sqrt{\xi^2 - 1}\right)\right)\omega_n\right)(v_i - v_{i-1}) \\ &- \left(C_1\left(\xi + \sqrt{\xi^2 - 1}\right)\omega_n\right)(v_i - v_0) \\ &- \omega_n^2(x_i - x_{i-1} + l_{i-1} + d_d) \end{aligned} \tag{4.2}$$

In Equation (4.2), u_i indicates the control input (i.e., the desired acceleration that should be sent to engine/brakes for actuation), a_i, v_i, x_i, l_i indicates the current acceleration, the speed, the position, and the length of vehicle i, respectively, while d_d indicates the desired inter-vehicle gap. C_1, ξ, and ω_n are control parameters regulating the weight between leading and preceding vehicle accelerations, the damping ratio, and controller bandwidth, respectively. The control algorithm considers data received from the leader and the preceding vehicle in the platoon, plus the gap to the preceding vehicle measured by the radar. This particular type of algorithms are defined as leader- and predecessor-following CACC and are proven to be string-stable under a *constant spacing* policy, which means that the inter-vehicle distance is fixed regardless of the cruising speed.

The work in [PSvN+11] instead defines a CACC that implements a predecessor-following control, meaning that a vehicle considers only information received by its predecessor. The control law, which is defined in terms of the derivative of the desired acceleration $(\dot{u})$, is the following:

$$\begin{aligned}\dot{u}_i = \frac{1}{H}(&-u_i + k_p\left(x_{i-1} - x_i - l_{i-1} - Hv_i\right) \\ &+ k_d\left(v_{i-1} - v_i - Ha_i\right) + u_{i-1})\end{aligned} \tag{4.3}$$

In Equation (4.3), which is defined as a Proportional Derivative (PD) controller, H indicates the time headway, while k_p and k_d are control gains for the proportional and the derivative part of the law, respectively. The control law has three components. The first one is the distance error (proportional term) and in this case the desired distance depends on speed (Hv_i). H is indeed the amount of time elapsing between two consecutive vehicles: the higher the speed, the higher the actual distance. This spacing policy is known as *constant time headway* and it guarantees string-stability for CACC considering the preceding vehicle information only: for these CACC string-stability under a constant gap cannot be guaranteed [Raj12]. The second one is the derivative of the first one (derivative term), which basically minimizes the speed error between consecutive vehicles. The last one (u_{i-1}) is the desired acceleration of the preceding vehicle. For the first two components of the law, the information about distance and relative speed can be obtained through the radar. The last one, instead, can only be obtained by means of communication, because the desired acceleration cannot be measured: it is an acceleration the vehicle will implement after a certain amount of delay due to actuation lags (engine and braking dynamics). This "knowledge of the future" enables to drastically reduce the time headway compared to a standard ACC while still guaranteeing string-stability.

In the literature, we find several other control techniques employed in the design of CACC. One example is Model Predictive Control (MPC) [KFF11]. This method solves an optimization problem on a future time horizon with the aim of minimizing certain quantities which include, for example, spacing and speed errors. Differently from the previous approaches, as this method relies on optimization, it is possible to define further constraints such as maximum and minimum acceleration for passengers' comfort. Without going too much into details, a typical MPC problem is defined as

$$\min_{\dot{\boldsymbol{u}}} \boldsymbol{z}^{\intercal} Q \boldsymbol{z} \tag{4.4}$$

subject to certain constraints, which include initial state, state evolution according to the applied control input, limits on acceleration and jerk, etc. In Equation (4.4), $\boldsymbol{z}$ might be defined as

$$\boldsymbol{z} = [\boldsymbol{e}, \boldsymbol{u}, \dot{\boldsymbol{u}}], \tag{4.5}$$

where $\boldsymbol{e}$, $\boldsymbol{u}$, and $\dot{\boldsymbol{u}}$ are the vectors of all the spacing errors, the control inputs, and the derivative of the control inputs over the prediction horizon, respectively. The matrix Q, instead, is used to weight the minimization terms. This problem is solved using standard mathematical solvers, with the result being a vector of jerk values (control input derivatives) $\dot{\boldsymbol{u}}$. Of this vector, the first value is the one being sent for actuation.

Still another type of approaches take a completely different perspective. While the majority of the control systems are defined in time domain, we find some approaches defined in space domain. As an example, the works in [BJ15, BJ17] define the spacing policy to be

$$x_i(t) = x_{i-1}(t - \Delta t). \tag{4.6}$$

The policy indicates that a vehicle should track a delayed version of the trajectory of its predecessor. The authors prove that this can be achieved if and only if all the vehicles are capable of tracking a reference speed signal defined in the spatial domain, i.e.,

$$v_i(x) = v_{i-1}(x) = \tilde{v}(x), \tag{4.7}$$

which is solved by defining a control law of the form $u_i(x)$, i.e., the acceleration a vehicle should apply depending on its position rather than the current time.

The list of approaches we mentioned is a minimal subset of the vast literature on the topic which include consensus control, event-triggered control, artificial potential field control, and many more [SSV+15, MSS+14, AGM15, TBJ17, SSV+19, SKVPA16, DPH17, GSBLC19]. This also includes the issue of lateral control, i.e., how vehicles should steer in order to follow their predecessors, for which the same string-stability property of longitudinal control must be guaranteed [KAFF14, SIS13]. The interested reader can refer to [DYW+16] for an in-depth view of CACC systems.

4.2 Platooning Coordination

This section is based on our paper [LBS^{+}21], in which we conduct a systematic literature review to analyze existing platooning coordination approaches. A systematic literature review was first introduced by [WW02] and consists of four steps: (i) search engine-based identification of possibly relevant literature using predefined keywords, (ii) filtering of the papers based on the title and abstract, (iii) detailed analysis of the approach sections of the papers, (iv) extraction of relevant approaches and classification into a taxonomy. We applied this technique for our literature review. In step 1, we use the search engine Google Scholar and predefined keywords to guide our search process. Thus, we use the following keywords for search: "platoon", "platooning", "platoon coordination", "platoon assignment", "platoon management", "platoon formation", "platoon formation strategy", "platoon algorithm", and "platoon multi objective". We consider the first 90-110 hits for each keyword, which results in a total of 1,630 papers. In step 2, we filter the papers based on the title and the preview in Google Scholar. For this filtering process, we rely on different assumptions to guarantee a consistent choice of literature. First, we do not restrict our analysis to a specific type of vehicle. Most of the approaches in the literature target platoons of trucks, as the potential for saving fuel seems to be the highest. However, we also include platoons composed of only cars as well as mixed platoons. Second, we focus on platooning coordination rather than platooning control. Hence, we excluded approaches that only target control aspects, such as communication/security mechanisms in platoons, inter-vehicular distance control, or string stability. Last, when focusing on platooning coordination, literature provides two types of approaches we include in our overview: (i) high-level approaches and (ii) focused approaches. High-level approaches mainly focus on assigning vehicles to platoons but support all phases of platooning. Focused approaches target specific aspects of platooning control, e.g., focusing on which position in a platoon a vehicle should join or the merge of platoons that drive close to each other. After this step, 155 papers are considered potentially relevant. In step 3, we perform a detailed analysis of those papers by assessing their abstract and parts of the sections that present the approach. In step 4, we extract the approaches of the remaining 40 papers.

4.2.1 Taxonomy

This section presents a taxonomy of platooning coordination composed of two categories `concept` and `strategy`. Both categories have several dimensions with various characteristics and handle individual conceptual levels. The concept category describes the basic design of the coordination approach. It describes which entity performs the coordination, which actions the coordination includes, and how the coordination is triggered in time. The strategy category describes the strategy that is applied within the presented concept, i.e., the exact way to determine coordination actions from an algorithmic perspective. Hence, the strategy category describes how to formulate the research issues defined using the aspects of the concept category. We use this distinction into concept and strategy to emphasize that—in theory—different strategies could be applied in the same concept. Figure 4.1 illustrates our defined taxonomy including the categories and their dimensions and Table 4.1 presents the characteristics for each dimension.

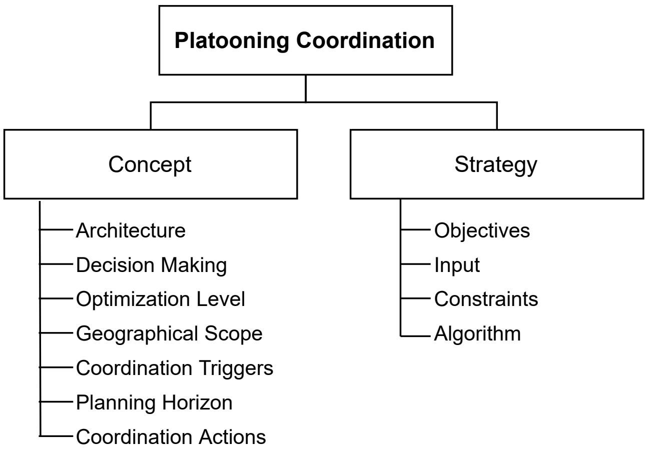


Figure 4.1: Illustration of the proposed taxonomy of platooning coordination approaches with eleven dimensions clustered into the two categories *concept* and *strategy*.

The concept category contains dimensions that describe the principle type of organization for the coordination procedure. In total, we identify seven dimensions: architecture, decision making, optimization level, geographical scope, coordination triggers, planning horizon, and coordination actions. First, the coordination process follows different fundamental design aspects. The architecture ranges from a central control unit for platooning coordination to decentralized coordination performed by each vehicle, including approaches in-between both extremes. The coordination processes can support different types of decision making w.r.t. the dimensionality of the decisions, i.e., the number of considered objectives. Further, those coordination decisions can target different optimization levels; we distinguish here between individual, platoon, and global. Additionally, platooning coordination has a spatial aspect.

Table 4.1: Overview of the taxonomy including both categories, their dimensions, and characteristics.

Category	Dimension	Possible Characteristics
Concept	Architecture	Control unit, individual vehicle, collective of vehicles
	Decision Making	Single-objective, multi-objective, many-objective
	Optimization Level	Global, platoon, individual
	Geographical Scope	Global, regional, local
	Coordination Triggers	Pre-trip coordination, event-based coordination, ongoing coordination
	Planning Horizon	Scheduled planning, real-time planning, opportunistic platooning
	Coordination Actions	Join, split, route, merge, leave, platoon parameter change, lane change
Strategy	Objectives	Energy efficiency, cost minimization, common distance, travel time, schedule miss penalties, comfort, road capacity, speed, profit, fleet utilization, failed actions, consistency of controllers, trust in leader, departure time
	Input	Destination, location, speed, time, route, vehicle characteristics, vehicles within horizon, road network, headway to vehicles, user preferences, fuel consumption models, departure, weather, traffic situation, platoon length
	Constraints	Deadlines, speed limit, platoon size, timing, speed, max. road capacity, lanes, distance
	Algorithm	Optimization-based, rule-based, heuristic-based, graph-based, iterative, greedy, game theory

The coordination could be employed by a central coordination unit that performs a global optimization. When mentioning a central coordination unit we refer to a central authority that is responsible for the decisions. Nevertheless, the calculation of the decisions can be distributed to several instances that exchange all existing information in order to handle the amount of instances. Still, the amount of required data might overwhelm the coordination process and heavily slow down the decision making. Hence, approaches may operate on different `geographical scopes`. Further, the coordination approaches differ in time aspects. The `coordination triggers` dimension refers to the triggers for the coordination process, i.e., the point in time when the coordination takes place. We distinguish pre-trip coordination (triggered by the driver before starting the journey), event-based coordination (triggered by events such as the split of a platoon during the journey), and on-going coordination triggered regularly. Related to that, we distinguish different `planning horizons`. Approaches either plan the whole journey in advance (scheduled planning), plan the rest of the journey based on real-time input (real-time planning) or spontaneously evaluate coordination actions without a particular planning horizon (opportunistic platooning). Last, coordination processes can support different types of `coordination actions`, e.g., joining a platoon, merging of platoons or adjusting parameters of a platoon (e.g., its speed).

While the first category depicts the general design of the coordination process, the `strategy` category focuses on how the coordination actions are determined. Hence, it describes how to formulate concrete research issues and contains the dimensions objectives, input, constraints, and algorithm. As captured in the dimension *decision making* of the concept category, platooning coordination follows one or several objectives. The dimension `objective` provides a list of possible objectives for the coordination. This list is based on [SKSB21]; however, we adjust the objectives to better reflect platooning coordination since the mentioned publication targets platooning in general. Obviously, one important characteristic of the platooning coordination strategy is the `input`. For instance, some strategies expect the specification of the route as a given input, while others calculate the best possible route as a part of the coordination strategy. Most of the platooning coordination approaches operate within specific `constraints`. These constraints include, for example, an intended speed range or specific vehicle types. Additionally, it is possible to define constraints to improve the coordination process, i.e., to avoid having unrealistic solutions (e.g., a speed of 300 km/h) or to limit the solution space for parameters to accelerate decision making. For making a strategy usable by ICT systems, we need to codify it in an `algorithm`. We identify several types of algorithms in the literature review, which use, e.g., optimization procedures, rules, heuristics or game theory.

4.2.2 Concept

In this section, we provide an overview of existing literature for the `concept` category. Therefore, we categorize them based on the dimensions and highlight the relevant details of the papers for this dimension[1]. The Figures 4.2 and 4.3 provide literature statistics on this classification.

[1] All analyzed approaches and their classification can be found at `https://doi.org/10.5281/zenodo.4267685`

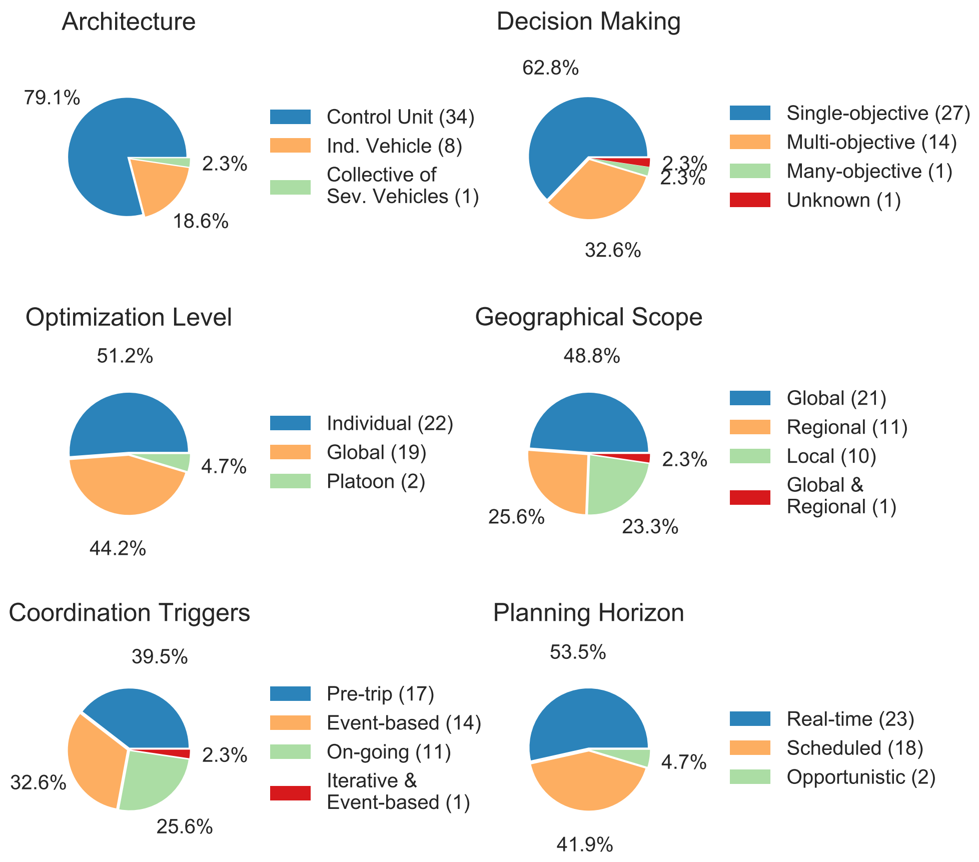


Figure 4.2: Results of the literature review on platooning coordination approaches. Relative and absolute frequency of the different platooning coordination characteristics in the 6 dimensions of the `concept` category.

4.2.2.1 Architecture

The majority of the approaches uses a dedicated *control unit* for platooning coordination. The control unit is able to make well-informed decisions for multiple vehicles simultaneously. Thus, this architecture is used for approaches that optimize a metric such as fuel efficiency or traffic flow on a whole highway, highway segment or across a region. For instance, [VJD15] propose an algorithm to find fuel-efficient routes and speeds for truck platoons before the start of their journey. A centralized control unit receives transport assignments by trucking companies and minimizes the aggregated fuel consumption of the whole fleet. In contrast, several approaches allow the *individual vehicle*

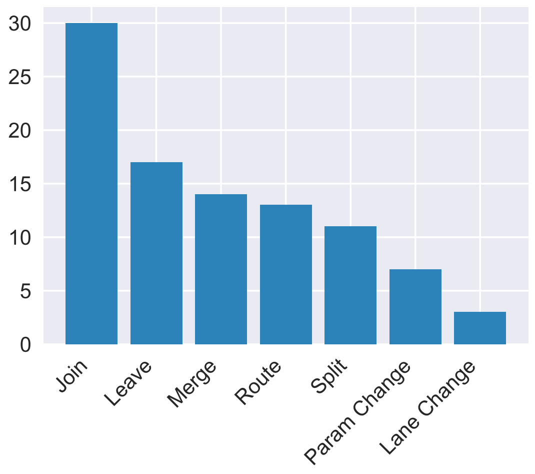


Figure 4.3: Results of the literature review on platooning coordination approaches. Absolute frequency of the actions that existing approaches are able to coordinate.

to plan actions. [HS19a] present an algorithm for platoon formation at urban intersections where vehicles communicate in a peer-to-peer fashion via beacons. Based on the transmitted positions of potential platoons, vehicles decide which platoon to join without communicating with any kind of control unit. We observe that—even though platooning is a cooperative driving technology—only [DCH08] introduce an approach where a *collective of several vehicles* coordinates the platooning process together. All vehicles on a certain highway segment collectively decide on platoon formation and lane assignment to maximize the distance that platoons stay intact.

4.2.2.2 Decision Making

We differentiate between *single-objective*, *multi-objective*, and *many-objective* decision making in platooning coordination. Single-objective approaches optimize a single parameter such as fuel efficiency or traffic flow under certain constraints. More than 60 percent of the approaches—including the large body of research on fuel-optimal route planning such as [VJD15, LSL15], and [VJD18]—apply this type of decision making. Multi-objective approaches optimize multiple, potentially conflicting objectives simultaneously. [MSH08]'s work from the KONVOI project is a prominent example for a multi-objective approach. They identify the savings through improved energy efficiency in platoons, the insurance savings through fewer accidents, and the wage costs for waiting times as relevant objectives. To balance these conflicting objectives, the approach combines them in a single profit function that is optimized by the coordination algorithm. Many-objective approaches also consider multiple objectives, but do not necessarily optimize all of them simultaneously. Based on user preferences or the current context, the approaches adjust the weighting of the objectives or choose, for instance, a different cost function. Many-objective optimization is rarely applied in platooning coordination research so far. Only [KB05] allow individual vehicles to maximize a custom utility function that may be different for each vehicle and that may change over time.

4.2.2.3 Optimization Level

Existing approaches either coordinate platoons to optimize their objective(s) on *global*, *platoon* or *individual* level. Global optimization encompasses approaches that maximize the objective for a larger group of vehicles. Thus, platooning coordination actions by these approaches may increase the fuel consumption of one vehicle to decrease the total fuel consumption of all vehicles in a highway segment. Often, such approaches still consider individual constraints of drivers. For instance, [LMS16] present a coordination approach that considers individual travel deadlines while minimizing the total fuel consumption of all vehicles. Only [BTV^{+}16] and [RLR18] describe approaches that optimize objectives on a platoon level. Much more common are approaches that choose the best action for each vehicle independently and thus optimize objectives on an individual level. [VWHK13a] present a system for each vehicle that autonomously decides to create, join or leave a platoon based on the current context and user preferences.

4.2.2.4 Geographical Scope

As far as the geographical scope is concerned, we distinguish between *global*, *regional*, and *local* platooning coordination. Global approaches coordinate all vehicles in the area under consideration and account for 49% of all approaches. A global geographical scope is especially prominent in routing approaches for truck platooning such as [VJD15] and [LSL15]. Regional coordination approaches divide the area under consideration in several regions that are managed separately. The coordination approach of the COMPANION project—presented in [LLJ15]—places control units at highway intersections. These control units manage the respective surrounding area by assigning approaching vehicles to platoons. The authors state the absence of a central unit which has sufficient knowledge and authority as well as the computational complexity of global optimizations as reasons for implementing a regional approach. Moreover, 23% of the approaches perform local coordination. These approaches coordinate vehicles in proximity without defining distinct regional borders. The majority of the local coordination approaches (9 out of 10) optimize the objective on an individual or platoon level, which shows that a local coordination is particularly useful when vehicles or platoons autonomously decide on their actions by considering only the situation in their immediate vicinity. [BTV+16] propose a three-layered approach for controlling each vehicle, a platoon or a fleet in one approach. Hereby, they perform both global and regional coordination.

4.2.2.5 Coordination Triggers

Platooning coordination approaches use *pre-trip coordination*, *event-based coordination* or *on-going coordination*. Pre-trip coordination happens before the start of the journey and is applied by 40% of all approaches. Drivers or trucking companies submit plans that contain, e.g., destination, deadline, and speed options. These plans are then leveraged by the platooning coordination approach to determine routes (e.g., in [SBH17, VJD15, NH16]) or coordinate platoon formation (e.g., in [MSH08, ZMJ16, SLM+17]). Event-based coordination approaches are triggered when certain events occur. An example for such an approach is [HC05]. The authors develop several heuristics for the formation of platoons on highway entry ramps. The coordination algorithm runs every time a new vehicle reaches the entry ramp and chooses an action such as joining a platoon, waiting for potential following vehicles or entering the highway. On-going platooning coordination approaches apply an algorithm periodically or even continuously. [BDSH13], for instance, periodically re-plan the current routes of platoons to optimize the traffic flow. Thus, the approach takes up-to-date traffic information into account and is able to react to unforeseen changes. The planning effort, however, is considerably higher and frequent plan changes may occur. [VMDJ19] propose an approach that works on-going and event-based. Monitoring and coordination run continuously while an additional event-based coordination is triggered in case of unexpected plan changes.

4.2.2.6 Planning Horizon

We observe different planning horizons in the platooning literature. Whereas some approaches plan the whole journey in advance, others evaluate platooning options spontaneously. We therefore distinguish *scheduled planning*, *real-time planning*, and *opportunistic platooning*. Scheduled planning takes place before the journey and plans the whole trip. It is used by 42% of the approaches. Scheduled planning is especially prominent in truck platooning where departure times, destinations, and deadlines are known in advance. Of the 18 approaches that apply scheduled planning, 15 work with trucks only and do not consider cars. Real-time planning organize the remaining journey while the vehicles are already on the road. More than half of all approaches (54%) use this type of planning horizon. The focus is on the formation of platoons on the road with some approaches even planning routes in real-time (e.g., [BDSH13, VMDJ19]). [VJD18] present a coordination approach that plans fuel-optimal routes for trucks. Whereas the core idea is similar to many scheduled planning approaches, the paper explicitly states that the proposed algorithm can be used during the journey if deviations occur or new information becomes available. Only two existing approaches, namely [KB05, FN15], apply opportunistic platooning. Instead of planning the remaining journey, the opportunistic approaches spontaneously evaluate platooning options. [FN15], present spacing policies for platoons to maximize the road capacity. Vehicles entering the road are able to spontaneously join platoons that drive nearby.

4.2.2.7 Coordination Actions

The broad range of platooning coordination algorithms in the literature is also reflected in the variety of actions that the individual approaches plan and coordinate. Figure 4.3 depicts the number of existing approaches that cover certain actions. Most platooning coordination approaches coordinate the platoon formation process by suggesting a suitable platoon for a new vehicle. About half of these approaches also coordinate the process of vehicles leaving the platoon to, for example, join another platoon or take a different route. The others do not explicitly advice vehicles to leave a platoon, but, for instance, assume that vehicles stay in the platoon until they reach their destination as in [HC05]. Apart from coordinating single vehicles to join or leave a platoon, several approaches are able to merge two or more platoons or split a platoon in multiple platoons. A considerable body of research also proposes routing algorithms for platoons, which is especially attractive for trucking companies with fixed deadlines. Platoon parameter changes or lane changes are only coordinated by a small subset of existing approaches. We observe that platooning coordination mostly happens on a high level of abstraction, leaving the execution of the suggested actions to a platooning control approach (cf. Section 4.1.2). Platoon parameter changes encompass the adjustment of a platoon's speed (coordinated by five approaches), inter-platoon positioning (two approaches), and inter- or intra-platoon spacing (two approaches).

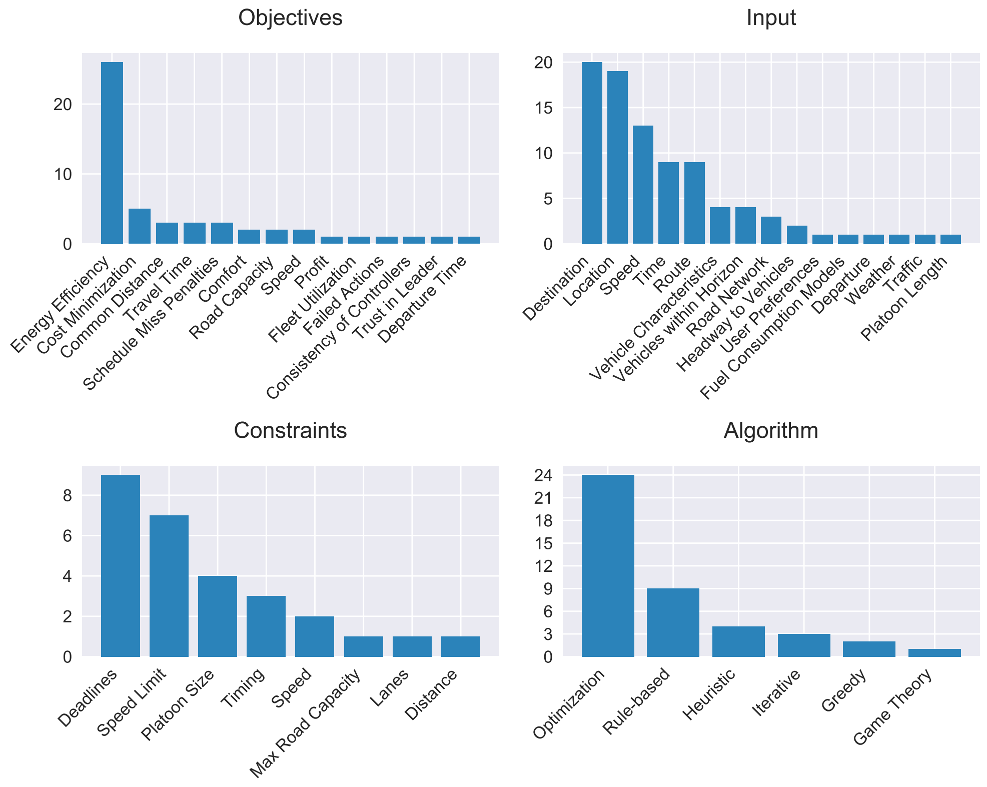


Figure 4.4: Results of the literature review on platooning coordination approaches. Overview of the `strategy` category with the objectives, inputs, algorithms, and constraints found in platooning coordination literature.

4.2.3 Strategy

Similar to the previous section, this section provides a broad overview of existing literature for all dimensions of the `strategy` category. Again, we decided not to discuss the papers individually but to categorize them based on the dimensions and highlight relevant details. Figure 4.4 provides literature statistics on this classification and we refer to it in the following sections.

4.2.3.1 Objectives

Our literature review reveals that energy efficiency is the dominating objective in platooning coordination. Closely related to this, five approaches minimize the overall cost of a journey which may—in addition to fuel costs—include wages [MSH08] or monetary penalties for missed deadlines [ZSK17]. Further objectives are only considered by three or less approaches. Several approaches define custom objectives that are only applicable to the respective approach. [VWHK13b], for instance, minimize failed actions. A feedback loop in the algorithm monitors which actions such as joining a platoon were successful. This metric is then used to evaluate different controllers that coordinate the driving process. A high consistency of these controllers is another objective mentioned in this work. The authors use different controllers in simulations, compare their consistency across several simulation runs, and prefer more consistent controllers.

4.2.3.2 Input

Input factors for a coordination strategy can be categorized in characteristics of the:

- drive (e.g., destination, current location or time deadline),
- user preferences (e.g., speed, route or optimization goals),
- vehicle characteristics (e.g., braking/acceleration coefficient, vehicle shape or fuel consumption model),
- platooning factors (intra-platoon position, time as leader or platoon size),
- traffic situation (e.g., traffic flow, headway to platoons or vehicles within horizon), and
- environment characteristics (e.g., road network, topography, weather or street condition).

We observe that information about the destination of a vehicle and its current location are the most frequent input for platooning coordination in the literature. This geographical information is not only relevant for routing approaches but also for a well-informed platoon formation decision that allows vehicles to stay in a certain platoon for as long as possible. Many approaches also require the speed of a vehicle, the time, and the route as an input. Surprisingly, we observe that information about the individual vehicle such as the vehicle characteristics (considered in four approaches) and the user preferences (one approach) are only used by a small subset of approaches. The same applies to important context information about the weather and the traffic situation that are both considered by only one existing approach.

4.2.3.3 Constraints

Platooning coordination strategies have to take constraints into account. The choice of the algorithms is limited by, e.g., the maximum size of a platoon or the maximum road capacity. In truck platooning, deadlines of transport assignments are frequently used as constraints. This ensures that, e.g., fuel-optimal routing does not lead to detours that prolong the journey excessively. In some approaches, e.g., in [Eil15, LMJ16], this is generically extended to more sophisticated timing constraints, which may include departure times, intermediate stops or rest periods. Several approaches limit the possible coordination actions by considering a speed limit. [Eil15] and [XSJ21] go beyond this and consider general speed constraints, for instance, in form of multiple, predefined speed options for each vehicle. [HS19a] formulate the constraint that the coordination algorithm only considers vehicles driving on the same lane as potential platooning partners.

4.2.3.4 Algorithm

The algorithm is the core of the coordination approach. It selects platooning actions such as joining or leaving based on the input information to achieve a certain objective under the specified constraints. Most approaches use optimization algorithms to evaluate different options. Here, (non-)linear programming (e.g., in [BDSH13, LSL15, VJD15]) is frequently applied to optimize an objective function. In addition, several approaches perform optimization with bio-inspired algorithms such as genetic algorithms (e.g., in [NH16]), ant colony optimization (e.g., in [NH19]) or grey wolf optimization (e.g., in [DRTR+19, DRTRS20]). The computational complexity of the optimization algorithms may render them infeasible for an application in practice. Fuel-optimal routing of vehicles, for instance, is proven to be NP-hard [LSL15]. Thus, several approaches operate based on rules or develop heuristics to eliminate the need for an exact optimization. Less common are iterative algorithms (all three approaches in [Lia14]) or the greedy choice of coordination actions [BTV+16,RLR18]. [JNJM18] apply a game-theoretic approach that models the strategic interactions between vehicles owned by different companies. This coordination approach assigns trucks with different departure times to platoons and shows that this cooperative behavior has its benefits.

4.3 Platooning Coordination Simulation

This section presents our simulation framework we published in [KLP+19]. The framework integrates the platooning simulator Platooning Extension for Veins (PLEXE) [SJB+14], which is based on Vehicles in Network Simulation (Veins) [SGD11] (including Simulation of Urban MObility (SUMO) and OMNeT++) with the Platooning Coordination System (PCS) for platooning coordination [KSB+18]. The simulation framework integrates two components for (i) a simplified definition of the platooning coordination strategies and the configuration of the simulation as well as (ii) a web interface based analysis of the simulation results. This enables user without experience in programming / simulation to use it.

Following the MAPE-K approach [KC03], the PCS is structured into the four key functionalities of monitoring the vehicles, analyzing which platooning actions are necessary (e.g., join requests of vehicles or inter-platoon actions), planning necessary actions, and controlling the execution of these adaptations. The monitor and the executor elements communicate with the vehicles (or the a platooning simulation) using a JSON-based protocol for collecting data and sending instructions, respectively. For analyzing and planning of the platooning behavior using the collected data, the PCS can integrate different coordination strategies to comply with individual objectives of drivers, e.g., travel as fuel-efficient as possible versus travel as fast as possible while keeping the benefits of platooning. These functions are supported by a shared knowledge repository that represents a typical self-adaptive systems approach [KRV+15] and enables adaptations of platooning behavior of vehicles at any time.

To simplify simulations with the PCS, we build a simulation framework connecting the PCS with PLEXE that includes a configuration tool and a web interface for the analysis of simulation runs. The PCS is well integrated into our simulation framework. Accordingly, users do not have to implement their strategies directly into the PCS to test new platooning coordination strategies. Users are rather able to just implement the algorithms of their strategies in a common Java class and export them as jar-files. We offer a development environment which supports an API for interacting with the PCS, such as accessing collected information or abstracting the commands of the JSON protocol. Accordingly, users can just implement their platooning coordination strategies, load them into the simulation tool and evaluate them.

We implemented the simulation tool using Python that invokes all required components of the simulation. A Graphical User Interface (GUI) permits the users to quickly setup a simulation, run it, and import the simulation outcome into a separate web interface for the visual analysis. Users can choose the PLEXE subsystem to be used, which can either be the version of PLEXE installed on the host machine or Instant-Plexe, i.e., a Linux-based Virtual Machine (VM), which comes with all the required software pre-installed. In addition, it permits the user to choose the installation location of the web analyzer for importing simulation results into the analysis tool.

Once the user chooses the subsystem to use, it presents a window enabling the quick creation of a simulation by entering some basic information such as the duration. The user can then choose the SUMO map and a traffic flow configuration. Currently, the tool offers two sample maps and a couple of pre-configured flow files, but the user can integrate additional ones. Next, the user chooses the coordination strategy. The different algorithms of the coordination strategies are loaded from a jar-file and displayed as a choice to the user. This will tell the PCS which coordination logic to load and to employ during the simulation. Finally, the tool creates a new PLEXE simulation folder including all the required configuration files. These files are standard OMNeT++ configuration files, as the ones included in any Veins or PLEXE tutorial. The experienced user can thus also generate a basic simulation and then manually tune additional parameters (e.g., IEEE 802.11p network parameters). The final tab permits the user to run a simulation and, at the end, to extract the data from the simulation and import it into the web analyzer tool.

The web interface for providing the graphical qualitative analysis of platooning coordination simulations was implemented using the Symfony Framework for PHP and JavaScript. It is composed of two web pages. The first one lists the simulations that have been run and permits either to analyze one or to choose two of them for comparison. The list of simulations is retrieved from a database which, in turn, is populated by the GUI interface. Once a simulation (or a pair) is chosen, the second page of the interface displays the SUMO map associated with the simulation and permits to re-play it. Each vehicle is displayed as a rectangle, and the color can either identify the vehicle type or the association to a platoon. Hence, vehicles belonging to the same platoon can be displayed using the same color. In addition, by clicking on a vehicle, the interface shows additional information such as *current speed* and its time evolution, *platoon id*, *position*, *distance* or *relative speed* to the front vehicle. Currently, we are developing a dashboard that shows the aggregated results of simulation runs w.r.t. platooning compositions, velocity, environmental pollution, energy consumption, and time spent in platoons.

Chapter 5

Logistics

According to the Council of Supply Chain Management Professionals, logistics can be defined as [oSCMP13, p.117]:

> "The process of planning, implementing, and controlling procedures for the efficient and effective transportation and storage of goods including services, and related information from the point of origin to the point of consumption for the purpose of conforming to customer requirements. This definition includes inbound, outbound, internal, and external movements."

This definition mentions two central processes in the field of logistics, namely the transportation and storage of goods. The transportation of goods refers not only to the actual transport, but also includes the planning of delivery routes. This planning process is mathematically defined as the well-known TSP and VRP. Warehousing of goods, in addition to actual storage, refers to other processes such as warehouse planning, storage assignment, and order picking. In addition to these two general processes, the logistics domain encompasses a broader variety of processes [MKS10, KT05, LR98]. While the concepts of SAS can be applied to many of these processes and could enhance the respective system, we put special focus on the processes of transportation and storage of goods. These processes are meaningful case studies for the application of our contributions as they are complex systems with high requirements and provide adaptation possibility. Still, the contributions of this thesis can also be applied to other use cases from the logistics domain for which the application of SAS appears meaningful. Further, the contributions can be generalized and applied also in other domains to which the application of SAS are meaningful such as, for example, in the packet routing within intelligent computer networks. However, the focus of this thesis is on the transportation and storage of goods, so this chapter only provides background information on the TSP and VRP, as well as on mezzanine warehouses.

5.1 Traveling Salesman Problem

The TSP is a highly researched optimization problem, first mentioned in the 1830s [Voi31]. It deals with the problem of finding the shortest possible route from a given initial city that visits all other cities exactly once and then returns to its initial city. This problem is known to be NP-hard and, hence, there is probably no algorithm that solves this problem in polynomial time.

The first philosophical mentions of the TSP are found in a literary magazine as a book advertisement in the 1830s [Voi31]. The advertised book describes the daily life of a traveling salesman and gives instructions on how to do the job, hints on good routes through Germany and Switzerland, and suggests places to stay. The first mathematical focus is found in the 1950s with the definition proposed by Merrill M. Flood [Flo56]. Later, it was implicitly proven that the TSP is NP-hard when Richard M. Karp proved the NP-hardness of the Hamiltonian cycle [Kar72]. The TSP has been and is still being extensively researched, as it can be used for a variety of real-world applications.

The TSP aims to find minimal routes within a network of cities and can therefore be represented by graphs. This graph consists of nodes representing cities and edges representing paths between nodes. The distances between nodes are represented by the weights of edges. The TSP is a minimization problem where the goal is to find a path that visits all existing nodes. This path must start from a particular node, which serves as the start and end node. In addition, the TSP can also be modeled mathematically by Integer Linear Program (ILP) formulations first proposed by Dantzig, Fulkerson, and Johnson [DFJ54] and Miller, Tucker, and Zemlin [MTZ60].

Exact approaches to solving the TSP include brute-force algorithms, branch-and-bound techniques [BT83], and linear programming [DFJ54,MTZ60]. Since the TSP is provably NP-hard, its complexity does not allow the computation of exact solutions for large problem spaces. Therefore, the focus has shifted to heuristic approaches that compute feasible solutions in a short time. David S. Johnson and Lyle A. McGeoch provide a comprehensive overview of heuristic approaches, including greedy algorithms, nearest neighbor, k-opt, Tabu Search, Simulated Annealing, GA, and Neural Networks, in their book chapter [JM03]. Besides, ACO are also commonly applied to the TSP [DG97].

5.2 Vehicle Routing Problem

The VRP is a generalization of the previously introduced TSP, which was first introduced in 1959 [DR59]. While the TSP considers a single vehicle for which a route must be planned, the VRP considers multiple vehicles for which routes must be planned taking into account a number of customers and orders. The goal is to reduce the total driving distance for all vehicles while serving all customers. Since it is a generalization of the NP-hard TSP, the VRP is also an NP-hard problem [Kar72]. In 1964, Geoff Clarke and John W. Wright proposed the first effective greedy heuristic for computing solutions for the VRP [CW64].

Similar to the TSP, the VRP can also be represented as a graph consisting of nodes and edges modeling customers and paths between customers, respectively. Again, the edge weight represents the distance between two nodes, and the goal is to minimize travel distances. However, unlike planning a single route with all nodes in the TSP, the VRP must compute a set of routes that consider all customers. The number of routes to be planned equals the number of vehicles. Three mathematical models for the VRP are proposed according to [LSG06]: (i) Vehicle flow formulation [Lap92], (ii) Commodity flow formulation [GG78], and (iii) set partitioning formulation [LSG06].

The survey by Gilbert Laporte and Yves Nobert categorizes exact solution approaches to the VRP into three groups [LN87]: (i) Direct tree search, (ii) dynamic programming, and (iii) ILP. However, the VRP is an NP-hard problem and, so there is no algorithm that finds an exact solution in polynomial time. This is the reason why researchers and practitioners use heuristic approaches such as Savings Algorithm, Tabu Search, Simulated Annealing, GA, ACO, and hybrid approaches [TV14].

The basic definition of the VRP is often extended by further constraints, which are necessary for the application to real use cases [TV14]. The capacitated VRP considers capacities of vehicles that must not be exceeded. This leads to the assumption of homogeneous or heterogeneous fleets, where in the homogeneous case all vehicles are assumed to have the same capacity, which is not the case in the heterogeneous instances. While the standard variant of VRP considers only one depot, the pickup and delivery variant (P&D) of VRP allows considering multiple pickup and delivery locations. This means that the delivery of a good must be scheduled with the same vehicle that picks up the good. This constraint leads to the consideration of multiple depots within one VRP, while the base version considers exactly one depot that is both the origin and destination. In addition, uncertainty can be part of the definition of VRP, since parts or all of the input may be unknown at the time of planning. This leads to dynamic VRP models that are able to handle uncertainty at design time [PGGM13].

5.3 Mezzanine Warehouses

Warehouses play a central role in the supply chain of a company and contribute to its logistical success. When employing humans, picker-to-parts and parts-to-picker methods are differentiated [DK07]. Experts estimate the picker-to-parts system to be the most common in Western Europe with a share of over 80% [DKLDR07]. A well-known picker-to-parts system is the mezzanine warehouse which we address in this work. Working within a mezzanine warehouse consists of two main tasks: (i) filling the storage with goods (storage assignment) and (ii) picking items out of the storage (order picking). The storage assignment problem defines the task of selecting storage locations to put a product into storage. The order picking problem defines the task of computing a pick route that collects the requested products of a customer order. Finding suitable storage allocations is important, as the allocation of products affects the travel distances during order picking. Due to the NP-hardness and, hence, the complexity of the storage assignment and the order picking problem, efficient optimization algorithms are required to find satisfying solutions within acceptable times. This section is based on our publication [LMK$^+$21b] and first introduces the mezzanine warehouse layout in Section 5.3.1 and presents state-of-the-art mechanisms for storage assignment and order picking in Section 5.3.2 and Section 5.3.3, respectively.

5.3.1 Warehouse Layout

Mezzanine warehouses usually store small-sized products that need to be picked by employees traveling through the warehouse. This type of warehouse consists of one or multiple floors to store goods using racks. Roodbergen and De Koster [RdK01] provide a general layout of such a mezzanine warehouse floor in their work from top-down view from which the following illustration in Figure 5.1 is derived.

Each floor consists of a predefined number of storage racks illustrated as white squares arranged in blocks, each consisting of three storage racks. The racks can be identified by their rack id depicted as the top number inside the rack, while the bottom number indicates the bay number. While the rack id uniquely identifies a rack within a floor, the bay number indicates the ordering of the racks within each block and, hence, is unique solely within a block. The blocks are separated by aisles of different sizes and directions: (i) horizontal cross aisles and (ii) pick aisles. While the horizontal cross aisles do not provide access to the racks, these are used to change the pick aisles from which the employee can access the racks. The cross aisles are wide enough to travel using

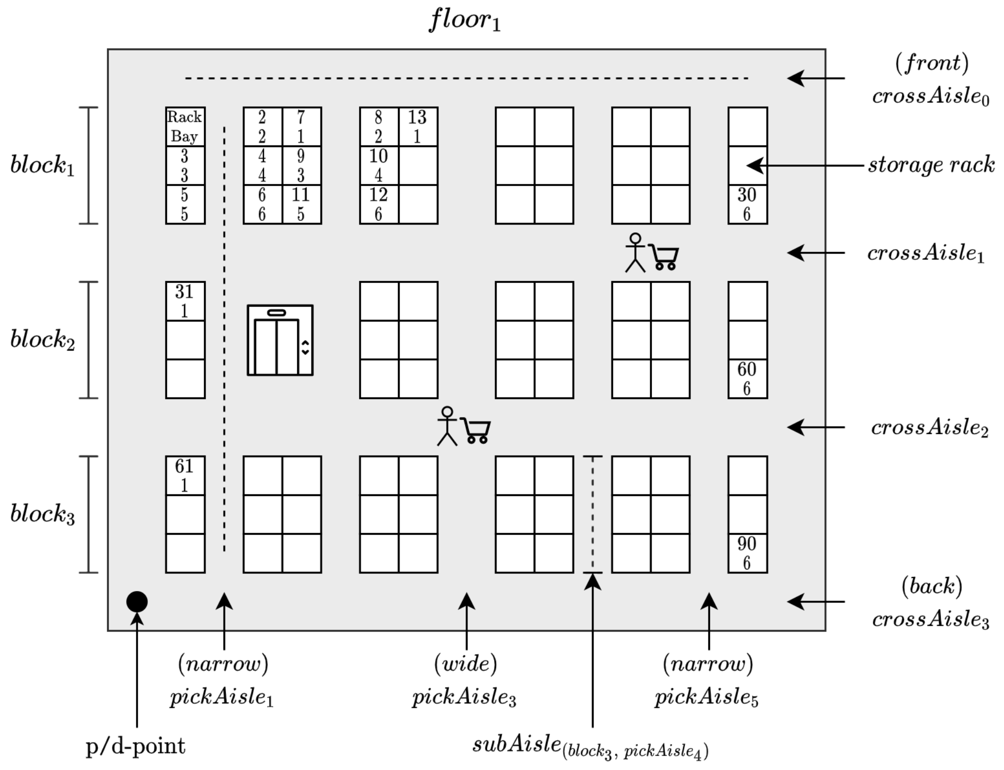


Figure 5.1: Example floor layout of a mezzanine warehouse (c.f. [RdK01]) consisting of blocks, storage racks, cross and pick aisles, and p/d-points.

picking carts. The pick aisles can be grouped into two types: narrow and wide pick aisles. Picking carts can be carried only in wide pick aisles and the employee needs to park the cart at a wide aisle to pick goods within narrow pick aisles. The part of an aisle within a block is called sub aisle of the according block, and, hence, a pick aisle consists of multiple sub aisles. The black dot at the left bottom of the figure indicates a p/d-point where employees need to deliver the picked goods or pickup the next set of goods to be stored in the warehouse. Finally, the employees are able to change the floor of the warehouse by using stairs or lifts.

The considered racks are identical in terms of their height, width, and depth within the warehouse. However, each rack can be configured individually with respect to the needs of the currently stored goods. Figure 5.2 shows possible configurations of the considered racks in this work depicted from the front. The rack configuration determines the number of shelves, that is, number of levels within a rack, and the number of compartments per shelf. Configuration 1 in the figure depicts three shelves each divided into two parts that represent compartments. Hence, this configuration offers six storage locations. Configuration 2 offers twelve storage locations by applying six shelves with two compartments each, and Configuration 3 offers 24 storage locations by dividing the six shelves into four compartments each.

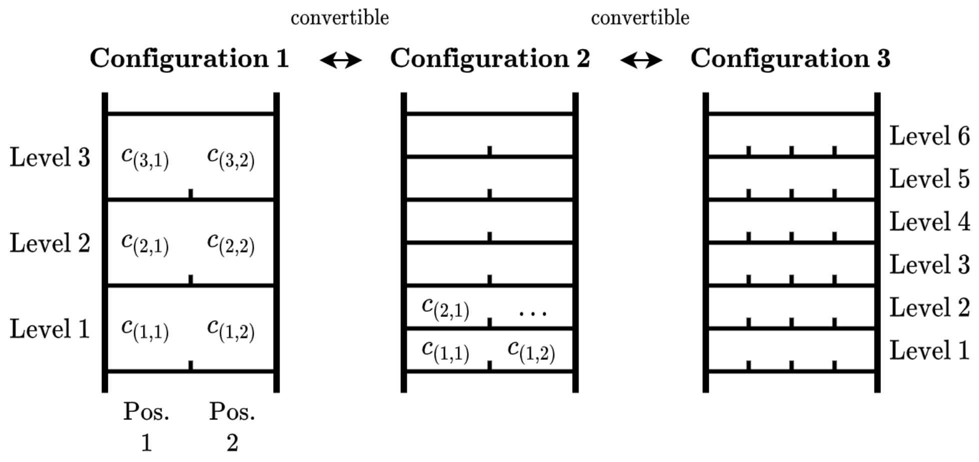


Figure 5.2: Example rack configurations that can be individualized with respect to the requirements of the stored goods. The configurations differ in the structure by the number of shelves as well as the number of compartments within a shelve.

5.3.2 Storage Assignment

The storage assignment problem defines the task of selecting storage locations to put a product into storage. Since mezzanine warehouses usually provide a large number of storage racks, it is difficult to find the optimum storage allocation that fulfills all custom constraints as well as ergonomic and economic objectives defined for the problem. The following sections introduce a subset of the most common storage assignment strategies present in the literature and applied in real-world warehouses.

The simplest storage policy is called *dedicated storage policy* [BIH19] in which each product is assigned to a dedicated and exclusive storage location. Using this policy, no changes in the warehouse need to be made and employees get to know the locations over time. However, the warehouse utilization is comparably low with a value of half of the storage capacity on average.

The *random storage policy* [BIH19] does not exclusively reserve locations for specific products but assigns incoming products randomly to unoccupied racks. This policy usually achieves better utilization values but comes with a higher administrative effort since the product locations change over time.

The *closest open location storage policy* [DKLDR07] reduces the randomness of the random storage policy by letting the employees select the storage location which results in selecting the first empty location the employee encounters. This leads to a higher utilization in the neighborhood of the p/d-points while racks that are farther tend to be empty.

The *rank-based storage policy* removes the randomness completely and ranks each incoming product based on a predefined rule set. These rules could contain but are not limited to [PSH05, Fra02, Hes63]: popularity, turnover, that is, the requested quantity by customers, the volume, the pick density, or the cube-per-order index.

Figure 5.3 illustrates four additional rank-based storage assignment policies introduced by [PS99]. This policy assigns storage locations based on the best- and worst-ranked products illustrated by black and white squares in the figure determined by their distance to the next p/d-point. The diagonal strategy (1) assigns incoming products according to their Euclidean distance to the next p/d-point, best ranked close and worst ranked further away from the p/d-point. The within-aisle strategy (2) assigns the best-ranked incoming products within the same aisle of the p/d-point. The across-aisle strategy (3) assigns the best-ranked incoming products to the entrance of all aisles that is nearest to the p/d-point. Finally, the perimeter strategy (4) assigns the best-ranked incoming products around the perimeter of the warehouse assuming that these are the most traveled aisles of the floor.

Class-based storage assignment strategies [DKLDR07] are a combination of the previously mentioned strategies as they include the following three tasks: (i) grouping of products into classes, (ii) definition of class regions within the warehouse, (iii) assign products to the defined region. Usually, the grouping is done using three classes (A-, B-, and C-class) that distinguish between fast- and slow-moving products.

Further, De Koster et al. [DKLDR07] take another factor into account for determining the optimal storage locations for products in their *family grouping strategy*. They include the correlation of products into account and propose to store products close to each other that often need to be picked in combination. They differentiate two types of strategies: complimentary-based and contact-

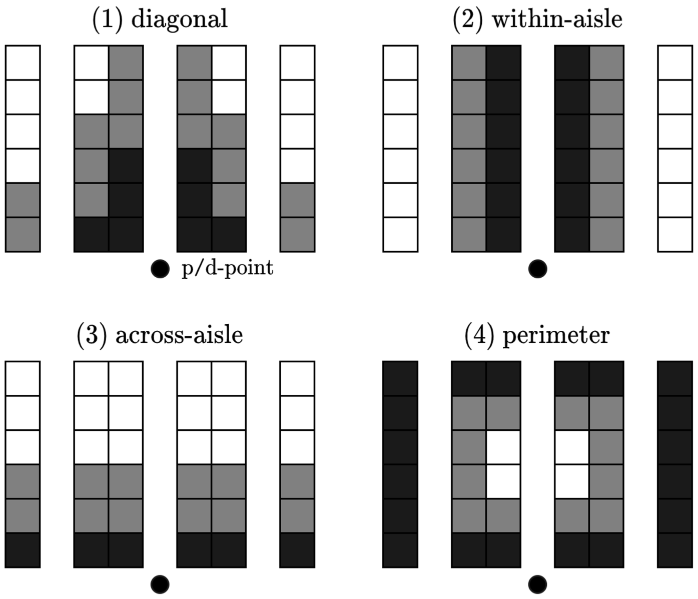


Figure 5.3: Illustration of rank-based storage assignment strategies (c.f. [PS99]). The storage locations are assigned based on the best- and worst-ranked products illustrated by black and white squares.

based, which measure joint demand or contact frequencies of the correlated products, respectively.

Besides the assignment of products to racks, the *golden zone assignment strategies* [PSH05] focus on the assignment of products into compartments. With golden zone these strategies refer to compartments located at grip height, that is between waist and shoulders of the picker. These strategies assign fast-moving products exclusively to the golden zones of racks and disregard the travel distance to the according rack.

5.3.3 Order Picking

The order picking problem defines the task of constructing pick routes within a warehouse that include all products of a pick list derived by a customer order. State-of-the-art routing heuristics are able to construct these routes fast and try to minimize the travel distance per route. They model the problem as TSP and start and end the route at a specific p/d-point.

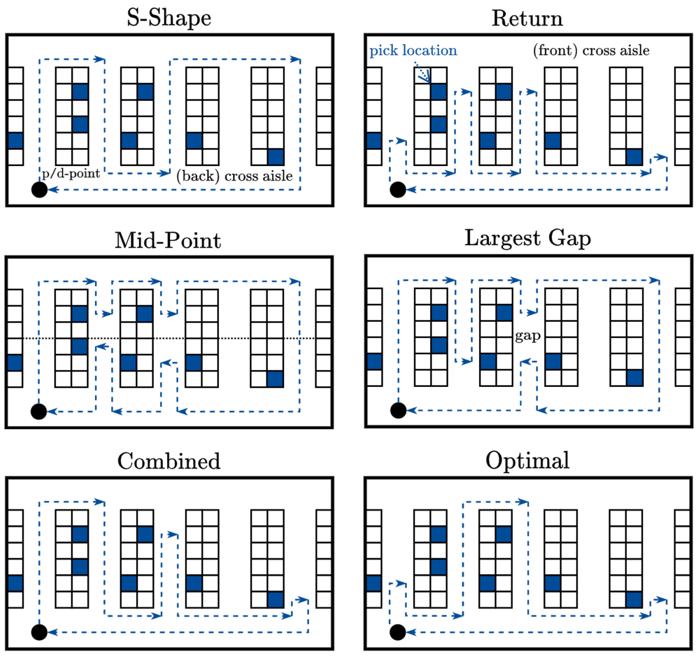


Figure 5.4: Illustration of five state-of-the-art order picking strategies and the optimal strategy based on dynamic programming (c.f. [Pet97, RR83]). Racks that contains items to be picked are marked in blue, the planned route is depicted as blue dotted line and starts at the p/d-point depicted as black dot.

Petersen [Pet97] proposes five routing strategies applicable for mezzanine warehouses and which are illustrated in Figure 5.4. The *S-Shape* strategy defines the route inside the warehouse to completely traverse all aisles that contain required products. It alternates the traversing direction so that a shape similar to the letter S is created as depicted in the figure. When applying the *Return* strategy, the picker enters the pick aisles in which required products are stored but always returns to the entrance of this aisle. Hence, in the worst case, an aisle might be traversed two times in case the product to be picked is stored in the last rack. In the *Mid-Point* strategy, the picker passes through each aisle at most to the middle of the aisle and then returns to the entrance through which he entered the aisle. In case a product is located further within the aisle, that is, behind the mid-point, the picker needs to enter the pick aisle from the other entrance again. The *Largest Gap* strategy adapts the idea of the Mid-Point strategy but dynamically sets the point that should not be traversed based on the largest gap between two products in the aisle. In this way, this strategy attempts to minimize travel distance by avoiding passing shelves that are not needed. Finally, the *Combined* strategy combines the S-Shape and Return strategies. It selects the pick aisle entry based on the current location of the picker and after all products are picked, the strategy decides to either complete this aisle and use the other entrance or to return to the initial entrance. In addition to these five strategies, Ratliff and Rosenthal [RR83] present another strategy called *Optimal* strategy where they apply dynamic programming to find the shortest route. For small problem instances, this method can be used to determine the optimum solution that can be used as gold-standard.

Chapter 6

Selected Motivating Scenarios

The central goal of this thesis is the design, prototypical implementation, and evaluation of a generally applicable self-aware optimization framework with a special focus on use cases in the ITS and logistics domain. The selection of these two domains was done due to their increasing research interest especially for the application of SAS to enable adaptability within a highly dynamic environment. Still, we are convinced that the motivating scenarios and the prototypical application of our contributions can be generalized and transfered to other domains besides ITS and logistics. In order to gain a deeper understanding of these use cases, we perform a set of case studies and further analyze important properties of the use cases. These case studies and analyses should be seen as motivating scenarios that will be further addressed in the course of this thesis and are one contribution of this thesis.

First, we analyze the form in which optimizations are useful in adaptation planning systems in order to define the general direction of the framework. Therefore, we use platooning coordination as representative for the ITS domain where the platooning coordination algorithm is the adaptation planning strategy that decides on adaptations of the entities in the system. Using this case study, we analyze in Section 6.1.1 how the adaptation planning strategy perform in different situations and how the current situation affects the optimal strategy and its parameterization. Our results show that the situation highly affects the adaptation planning strategy performance and that the strategy should be dynamically selected and tuned to the current state of the system and the environment.

Second, motivated by the findings of the literature on the advantages and disadvantages of platooning, such as the lower slipstream benefits for vehicles at the front (c.f. Section 4.1, [BSC+12]), we analyze fairness aspects for platooning participants. Therefore, we performed a case study in Section 6.1.2 on adaptation planning strategies that manage the sequence of vehicles of a platoon. We designed six rotation strategies for platoons with the aim to equally distribute benefits and disadvantages while driving in a platoon. Our study utilizes different platoon sizes, platoon speeds, as well as traffic and road conditions to analyze how well they distribute the benefits and which mechanism performs best in which situation. The results show that all strategies distribute the benefits but have their individual sweet spot with respect to the traffic and road conditions.

Third, as the targeted use cases operate in dynamic environments to which they need to adapt, uncertainty is another important characteristic. In our third case study in Section 6.1.3 we analyze aspects of a proactive adaptation planning system for route planning. We study uncertainty in the use case of fuel price volatility and integrate uncertainty aspects into utility functions. Our case study shows that uncertainty is an important property of the targeted use cases and need to be addressed when applying adaptation planning strategies.

Analogous to the ITS domain, we argue that these properties can also be transferred to the logistics domain. We discuss two further use cases called (i) storage assignment and order picking in Section 6.2.1, and (ii) vehicle routing in Section 5.2 for this domain and highlight important challenges these use cases add to this thesis. The common challenge of both is their systems-of-systems structure and, hence, the inter-relatedness of the handled problem statements. While we classify the storage assignment and order picking problem as horizontal systems-of-systems, we call the VRP a vertical systems-of-systems.

6.1 Intelligent Transportation Systems

The ITS domain is the first domain we address in this thesis including three case studies on platooning coordination, platoon vehicle sequence, and route planning. All use cases aim at a specific challenge faced by adaptation planning strategies in dynamic environments. Nevertheless, all challenges can be found in all mentioned use cases. We first analyze the situation-dependency of adaptation planning strategies to motivate the meaningfulness of our framework on the platooning coordination use case. Afterward, we study fairness aspects within the platoon as the benefits and disadvantages of platooning are not equally distributed among all participants. Finally, we discuss the uncertainty challenge in highly dynamic environments in the example of route planning with volatile fuel prices.

6.1.1 Situation-dependency

To analyze the meaningfulness of a self-aware optimization framework, we conducted a study on the situational behavior of platooning coordination strategies as a representative of ITS and presented the results in [LNH$^+$21]. This section aims to motivate the self-aware optimization framework by presenting the study results. Therefore, we provide only the most important details of the study design and omit some details as they would unnecessarily lengthen this section. Nevertheless, all details of the study can be found in our paper [LNH$^+$21].

In the aforementioned study, we analyzed the situation dependence of three platooning coordination strategies by applying them to twenty different traffic scenarios and optimizing their input parameters using four optimization techniques. We define four baseline traffic scenarios with the parameters platoon percentage, car spawning rate (car/h), truck spawning rate (truck/h), and speed limit (km/h). Then, we combine these base scenarios with five variations using the parameters maximum platoon size, allowed overtaking, and number of lanes. This combination results in a total of 20 traffic scenarios.

For the coordination of platoons, we use the following three well-known platooning coordination strategies [KLP$^+$19]: (i) best velocity, (ii) closest distance, and (iii) closest distance and lane. The best velocity strategy defines the best matching platoon by calculating the velocity difference between platoon and vehicle and selecting the platoon with the lowest positive speed delta. The closest distance strategy analyzes the distance between vehicle and possible platoons and selects the platoon with the lowest longitudinal distance. The closest distance and lane strategy not only calculates the longitudinal distance of vehicle and platoon but incorporates the number of lanes between them.

We use four optimization techniques to adapt the input parameters of the platooning coordination strategies to the current traffic situation: (i) NSGA-II [DPAM02], (ii) a modified NSGA-II version using Novelty Search [LS10], (iii) Bayesian Optimization [BCDF10], and (iv) Simulated Annealing [KGV83].

As target metrics for the optimization algorithms, we define four objectives with corresponding metrics to analyze the currently selected input parameters' performance. The metric *throughput* analyzes the number of arriving vehicles per hour compared to the number of starting vehicles per hour and covers platooning goals for traffic flow and road capacity. The *time loss* metric calculates the average time in seconds lost by all vehicles on the road compared to their expected travel time and covers velocity and time goals of platooning and vehicle-specific objectives for platooning coordination. The *platoon utilization* metric measures the average platoon size, that is, the average number of vehicles in the platoon, compared to the maximum platoon size and covers platoon-specific goals of platooning coordination. Finally, the *platoon time* metric compares the time traveled in a platoon in seconds to the total travel time and covers time and user comfort metrics of platooning.

6.1.1.1 Situation-Dependent Behavior of the Coordination Strategies

In this experiment, we compare the strategies in 20 traffic scenarios and analyze, which strategy optimizes which metric as displayed in Figure 6.1. The x-axis displays the different platooning coordination strategies; the y-axis shows the number of best solutions each strategy has for each metric. Since 20 different traffic situations exist, a coordination strategy can have 20 best solutions at maximum. To have the best solution for a given metric and traffic situation, a coordination strategy needs the best score for the metric and scenario variation. As can be seen, none of the strategies performs best for each metric. Consequently, the relevant metrics drive the choice of the platooning coordination strategy and, hence, this choice is objective-dependent. Further, as each strategy optimizes the throughput metric in a specific scenario, even for a dedicated metric, a specific strategy might be superior in a specific traffic scenario. Therefore, the choice is also situation-dependent which leads to the conclusion that adapting to the traffic situation by switching the strategy is beneficial.

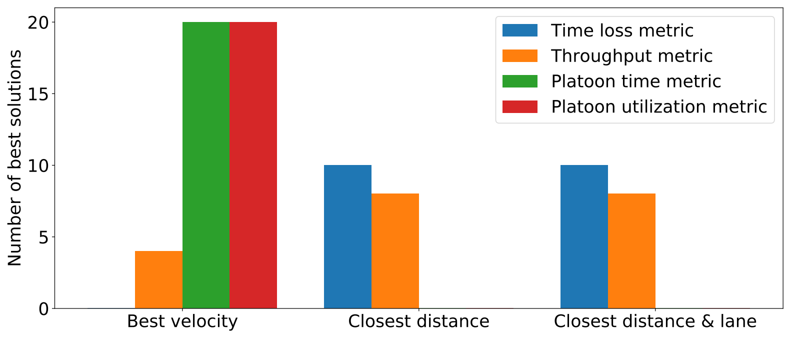


Figure 6.1: Comparison of the three platooning coordination strategies over all 20 traffic situations. The metrics are depicted as blue bars representing the number of best solutions identified by the particular strategy.

6.1.1.2 Situation-Dependent Behavior of the Strategy Configuration

We now focus on one specific strategy and analyze the relevance to take various configurations into account. The experiment focuses on the metric platoon utilization and the best velocity strategy as this strategy shows the best solutions in all of the 20 traffic scenarios. We define three configurations for the best velocity strategy and group the traffic scenarios by the base scenarios. Each of the three defined strategy configurations might be superior depending on the traffic condition. Figure 6.2 shows how many best solutions each configuration of the best velocity strategy has for every scenario. The maximum number of best solutions per scenario is five. The figure shows that Configuration 2 only has the best solutions in Scenario 2 and 3. The other configurations have best solutions in every scenario. Furthermore, there is no scenario with only one configuration having the best platoon utilization solution for every variation. Consequently, the results demonstrate the benefit of adapting the platooning coordination strategy's configuration to the specifics of the current traffic situations as even slight changes in the conditions need to be coped in the parameters.

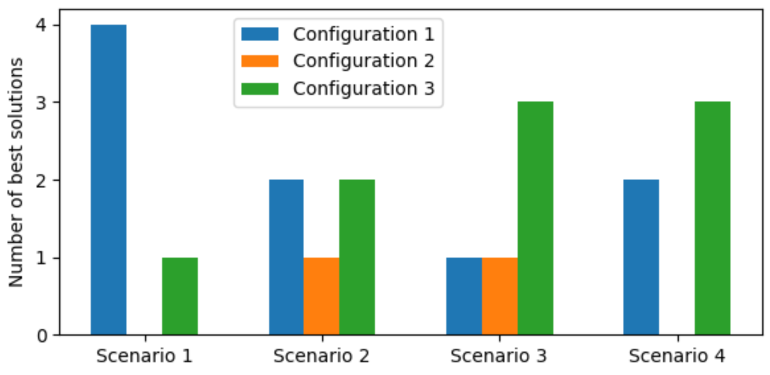


Figure 6.2: Analysis of the velocity based strategy for the four base scenarios. The three different configurations are depicted as colored bars which represent the number of best solutions identified by the configuration in each scenario.

6.1.1.3 Unique Contribution of the Optimization Algorithms

First, we compare the optimization algorithms using only the concept of Pareto dominance. Figure 6.3 combines the solutions of all Pareto fronts produced by every optimization algorithm into one front. First, it displays the size of the final Pareto fronts in green. Additionally, the plot contains the number of contributions and unique contributions for every optimization algorithm. The contribution indicates the number of solutions in the final front produced by an optimization algorithm which are not dominated by solutions in the combined front. A unique contribution is a contribution that was only made by one optimization algorithm. Ranking the optimization techniques, Bayes is first in terms of contributions, followed by Simulated Annealing and NSGA-II, which provide comparable results, and finally Novelty Search with low amounts of contributions. However, every optimization technique—except for

Novelty Search with a novelty weight of 0.3—contributes unique solutions to the combined Pareto front. Thus, no algorithm dominates another one entirely.

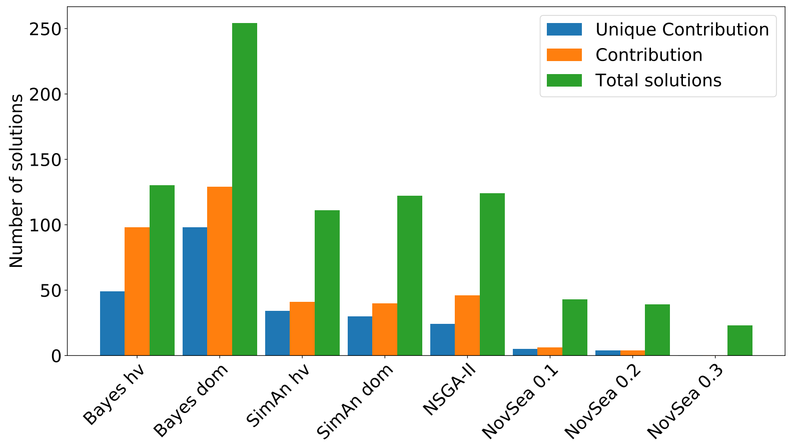


Figure 6.3: Analysis of the contribution to the best-known front for each optimization algorithm. The colors represent the number of total solutions, contributions and unique contributions per algorithm.

6.1.1.4 Objective Score Dependent Performance of the Optimization Algorithms

Now, we analyze the final Pareto fronts of every optimization algorithm based on the objective scores as summarized in Table 6.1. The table highlights the best average value per objective. Taking all platooning coordination strategies into account, the evaluation calculates the average and standard deviation over the final fronts of all strategies. Overall, Bayes has great results for every metric, especially when taking both algorithm versions into account. NSGA-II and Simulated Annealing show average values overall. Novelty Search is below average, except for the platoon utilization metric of versions 0.2 and 0.3 as well as the time loss metric for Novelty Search 0.1. Furthermore, there is a tendency for the HV and dominance rank versions of Simulated Annealing and Bayes. The solutions produced by the dominance rank show high scores for time loss and throughput while the HV versions dominate the platooning-specific metrics. The HV based approaches may focus on more fluctuating metrics since a greater increase in metric score leads to better HV values. This complies with the findings in this table, showing a higher standard deviation in the platoon metrics. Additionally, the high standard deviation of Bayes with dominance rank is another interesting aspect. The observations in Figure 6.3 show a high Pareto front size of Bayes with dominance rank combined with higher standard deviation indicate that this optimization algorithm finds more diverse solutions providing opportunity for adaption to user preferences.

Table 6.1: Average objective score and standard deviation for every optimization algorithm and objective.

	Time loss		Throughput		Platoon time		Platoon util.	
	mean	std	mean	std	mean	std	mean	std
Bayes hv	0.887	0.015	0.732	0.007	**0.475**	0.159	**0.572**	0.115
Bayes dom	**0.907**	0.016	**0.737**	0.005	0.402	0.175	0.486	0.128
SimAn hv	0.890	0.014	0.733	0.007	0.461	0.142	0.559	0.087
SimAn dom	0.902	0.013	0.736	0.006	0.400	0.140	0.505	0.104
NSGA-II	0.905	0.015	**0.737**	0.006	0.390	0.137	0.502	0.124
NovSea 0.1	0.900	0.014	0.734	0.006	0.379	0.132	0.500	0.114
NovSea 0.2	0.897	0.014	0.734	0.006	0.396	0.143	0.517	0.121
NovSea 0.3	0.893	0.013	0.732	0.006	0.405	0.127	0.526	0.105
Default	0.897	0.009	0.735	0.003	0.363	0.033	0.494	0.034

6.1.1.5 Relation of Optimization, Adaptation Planning, and Objectives

The following Figure 6.4 contains boxplots for showing the relation between optimization algorithms, platooning coordination strategies, as well as objectives. As it can be seen, to optimize different objectives, not only a specific planning strategy is superior, also for the optimization different algorithms might improve the planning strategies differently. As one specific example, for the objective platooning utilization, it can be seen that for the strategy best velocity NSGA-II performs best. However, it performs worst for the closest distance; contrary behaves Simulated Annealing. Also, it can be seen that for a specific planning strategy, not a single optimization algorithm performs best. For example, for the closest distance and lane strategy, the Simulated Annealing algorithm performs worst when targeting the objective time loss. However, it is superior for the platoon utilization objective. Finally, it can be concluded that the choice of the adaptation planning strategy but also the optimizer to improve the strategy's parameters is not a "one fitting all" choice, especially in multi-objective scenarios. Still, it can be visible that in most scenarios, the application of any of the three considered optimization techniques still outperforms the default parameters of the adaptation planning strategies' default parameters, which has been already defined by domain experts.

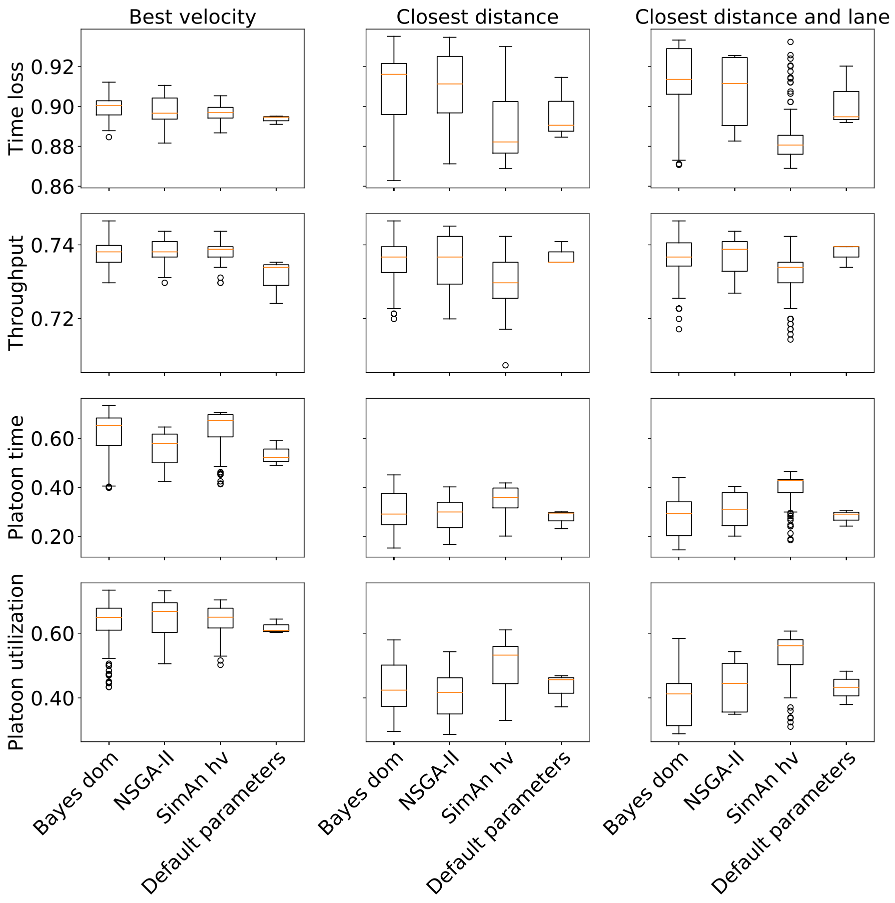


Figure 6.4: Comparison of optimized and default solutions for platooning coordination strategies shown in the columns for the four metrics shown in the rows.

6.1.2 Fairness

As a second aspect, a self-aware optimization framework should address, we propose to incorporate the fairness of adaptation actions. Similar to Rescher, we see fairness as "dividing goods or bads on the basis of general principles that pertain to everyone alike" [Res02, p.13]. We again analyze fairness in the platooning domain as one example of ITS. Vehicles driving in a platoon benefit from slipstream effects due to air drag reduction resulting in energy savings [Seg16, Joo12, BSC+12]. Additionally, the global traffic flow is optimized by the homogenization of velocities as well as increased traffic throughput [BSC+12]. However, the vehicles in such a platoon experience unequal benefits depending on their position in the platoon. Especially the lead vehicle experiences reduced fuel savings of around 5% while follower vehicles experience fuel savings of up to 15% [BSC+12]. Additionally, in some approaches, the driver of the lead vehicle has to drive manually whereas other vehicles can follow in a self-driving mode.

In [LKS+21], we analyze compensation-centered incentives for platooning to enable the platooning technology to become more fair and incentivize vehicles to take part of it. We propose a taxonomy of these incentives and further design six indirect compensation mechanisms. These mechanisms base on the idea to equally distribute negative effects of platooning similar to the fairness definition of Rescher [Res02] to create a more fair platoon composition and hence, aim at a broader acceptance of platooning in the real-world.

As the benefits and drawbacks in the platooning domain depend on slipstream effects, wind and/or vehicle model, and all these impact factors cannot be simulated within one simulation, we focus on the intra-platoon position each vehicle takes during the trip. Hence, we track the time driven at a certain position inside the platoon relative to the time overall spent in the platoon, for all cars belonging to a platoon. Ideally, to distribute benefits and drawbacks equally over the platoon members, every vehicle should have spent an equal amount of time in the leading and tail position respectively.

As already mentioned, we design six mechanisms to compensate for less positive effects in platooning: (i) Drafting a Single Vehicle to the Front (DtF), (ii) Drafting a Single Vehicle to the Back (DtB), (iii) Belgian Tourniquet (BT), (iv) Belgian Tourniquet Jump-start (BTJS), (v) Reversed Belgian Tourniquet (RBT), and (vi) Reversed Belgian Tourniquet Jump-start (RBTJS). All mechanisms are designed to be run during the whole journey within a platoon, that is, whenever one round of the mechanism has finished, the next round starts immediately. In DtF, one of the following vehicles, for instance, the last vehicle of the platoon, overtakes the platoon and takes over the lead. In DtB, the leading vehicle will leave the platoon, switch lanes and then let the platoon pass before queueing up behind it as the new tail of the platoon. The BT mechanism originates from professional cycling, where the platoon is separated into two groups across two lanes and a constant rotation takes place. The mechanism is similar to DtF, however, the next vehicle starting an overtake is always the

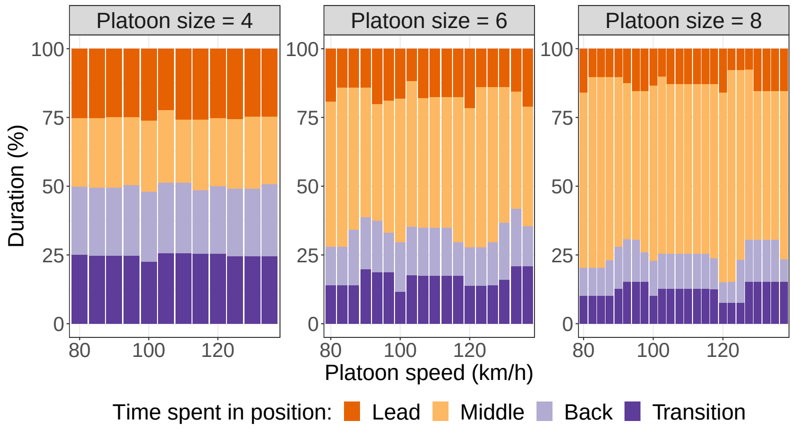


Figure 6.5: Time spent in different intra-platoon positions for each platooning vehicle with regards to three different platoon sizes in the three plots shown for DtF. The different colors represent the positions in the platoon and the stacked bars per platoon speed represent the individual vehicles.

last one of the original platoon. In the jump-started version of BT, called BTJS, uses a modified start-sequence for the rotation: The trailing half of the platoon switches lane synchronously once enough space is available to jump-start the rotation and falls back to queue up behind the platoon. In the reversed version of BT, that is RBT, the leading vehicle is next to switch lanes and fall back once enough space is provided to queue up behind the platoon. Similar to BTJS, we also examine an alternative version to the RBT, jump-started by the preceding half of the platoon switching lane simultaneously.

First, we discuss the fairness of our proposed mechanisms, i.e., how well the compensation inside the integrated subsystem performs, by inspecting the time each platoon vehicle spends in the different intra-platoon positions. Since we want to analyze the behavior of the isolated methods and do not want to measure the influence of the surrounding traffic, lane count, and speed limits have on our methods, we fix these parameters to: no traffic, three lanes, and no speed limit. Figures 6.5 and 6.6 depict the time spent in the according position inside the platoon for each vehicle in percent. The x-axes show the platoon speed and a vertical bar for each vehicle is depicted for each speed scenario. The y-axes depict the duration in seconds. The colors represent the different positions inside the platoon, where orange depicts the time at the lead, yellow the time inside the platoon between the leader and back vehicle, that is presented in light blue, and the time spent for overtaking or falling back, i.e., during the transition, is depicted in dark blue.

Figure 6.5 compares the time spent in the different positions using the DtF mechanism for different platoon sizes, four vehicles on the left, six vehicles in the middle, and eight vehicles on the right. For all scenarios, it can be seen that all bars for each platoon speed and platoon size show similar color distributions, i.e., the time in the various intra-platoon positions is spread equally among all vehicles of the platoon. The small deviations between single vehicles inside a platoon can be explained by the timing the simulation has ended, as can be retraced due to the consistent ordering of bars. For example, for a platoon size of six and a velocity of 100 km/h, the vehicles reached the end of the road while the vehicle starting in the second position was overtaking or has only freshly taken over the lead. Thus, all vehicles have similar times in each intra-platoon position regardless of the initial position at platoon formation time. For a platoon size of four, it is also visible how the travel time of each vehicle splits into four equally sized portions representing the different positions. The reason for this is that a new overtaking maneuver is started when the previous ended. Hence, one vehicle is situated in transition at all times. With increasing platoon sizes, this pattern is not present anymore because now multiple cars are positioned between the leader and tail of the platoon. Thus, the larger the platoon size, the longer the time in the favorable middle of the platoon while the position with negative effects are distributed equally over all participating vehicles of one platoon.

Figure 6.6 shows the same kind of diagrams for all six mechanisms but a platoon size of eight only. We decided to show the results for a platoon of eight vehicles as the characteristic effects of the different intra-platoon positions become most visible there. When comparing the DtF and DtB, the figure shows that also in the DtF method, all eight bars for each platoon speed show nearly the same color distribution, i.e., all vehicles have similar times in each position regardless of the initial platoon formation. The pattern of this distribution is similar to the one for DtF and the cars are situated in the middle, i.e., the most favorable position, most of the time. The patterns of color distribution of the BT variants differ from the ones before, as the dark blue bars are significantly higher. This means that the cars spend more time in transition. As this lies in the nature of the BT, this shows that the desired effects occur and constant rotation is performed. However, when a vehicle is not in transition, it still drives most of the time in the middle of the platoon. Additionally, a notable difference can be seen for the first and the last four cars of the platoon, as the first car is the leader of the platoon for a longer time and the last vehicles have increasing time at the back and in transition.

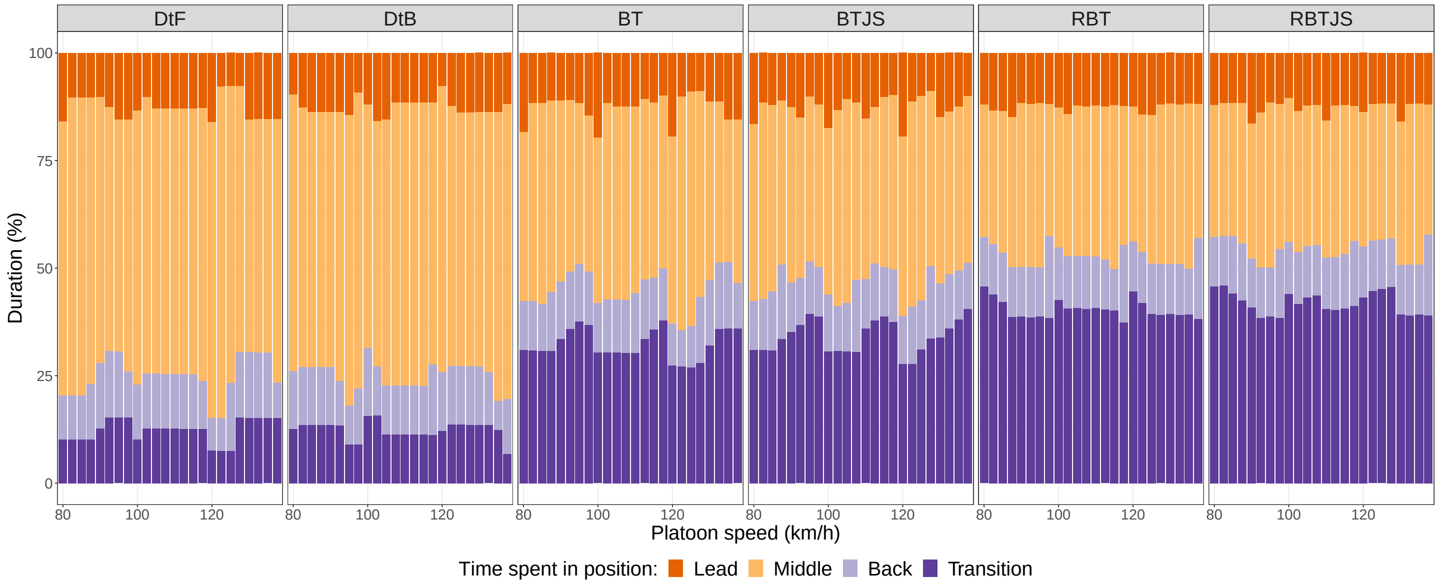


Figure 6.6: Time spent in different intra-platoon positions for each platooning vehicle shown for all methods and a platoon size of eight. The different colors represent the positions in the platoon and the stacked bars per platoon speed represent the individual vehicles.

This can be explained by the characteristics of the BT as it requires some time to get the rotation going. This effect can be explained by the overtaking platoon that needs to be formed on the left lane from scratch at the beginning of the rotation. The jump-started version of this mechanism shows similar behavior. However, more vehicles have the effect of increasing time in the back and transition, as multiple cars switch to the left lane for overtaking at the same time in the beginning. In the reversed versions of the BT, the cars spend more time in transition than in all other mechanisms. Besides, the time in transition decreases with regards to the initial position in the platoon, so that the first car spends most of the time in transition and the last car the least. This lies in the nature of these mechanisms as the fallback procedure requires more time for the fallback as we implemented the mechanisms to minimize the disturbance of other traffic. However, these mechanisms distribute the time as leader equally among all cars and no negative effects happen for the initial leader as in the other two BT versions. In summary, for the DtF and DtB mechanisms, the time spent in positions with negative effects is split equally among all vehicles and they drive in the middle of the platoon most of the time. The BT versions show negative effects for the initial leader and the cars drive more time in transition, while the reversed BT mechanisms do not show the negative effects for the initial leader but the cars spend the most time in transition.

6.1.3 Uncertainty

Additionally to situation-dependency and fairness aspects of adaptation planning, uncertainty is a further important aspect for this work. Ramirez, Johnson, and Cheng define the term uncertainty in the context of Dynamically Adaptive System (DAS) as follows [RJC12, p.101]:

> "Uncertainty is a system state of incomplete or inconsistent knowledge such that it is not possible for a DAS to know which of two or more alternative environmental or system configurations hold at a specific point. "

According to Esfahani and Malek "Uncertainty can be observed in every facet of adaptation, albeit at varying degrees" [EM13, p.2]. Mahdavi-Hezavehi, Avgeriou, and Weyns describe a set of aspects that induce uncertainty into self-adaptive systems such as "the dynamicity and unpredictability of a variety of factors existing in software systems" [MHAW16, p.4] as well as "consequences of self-adaptation in a software system" [MHAW16, p.4]. Hence, we conducted a third study in the domain of ITS and analyzed uncertainty aspects of a proactive adaptation planning system that plans routes and incorporates refueling if required [LHKK21b]. Bousonville et al. [BHMK11] identified that German fuel prices vary across geographical regions. Additionally, it can be observed that fuel prices in some countries are highly volatile throughout one day.We study those claims in CostSaVeR [LHKK21b], a self-adaptive navigation system that applies multi-criteria optimization for cost-aware routing. The application includes a decision logic to analyze possible routes using utility functions and proposes the best found route option. Using our evaluation testbed, we assess the performance of our utility functions as well as the importance of reasoning on the uncertainty of price stability with travel distance or time.

As already mentioned, we introduce six different utility functions as a representative set of three categories: (i) integrating measures of the gas price, distance, and duration; (ii) coping with the uncertainty of volatile gas prices; and (iii) selecting either the nearest or completely random gas stations. The *price-aware* utility function summarizes the already paid costs of the remaining amount of fuel and the new refueling costs for the remainder of the journey. The *duration-/distance-aware* utility function (Dur/Dist) calculates a weighted sum of the duration and distance of the journey into one utility value. Based on the assumption that the probability of a change of the price at a gas station increases with a larger distance towards the gas station, the idea of the *volatility-aware* utility function is to reward closer gas stations with a utility bonus. Therefore, the price-aware utility function is calculated and a bonus is added concerning the distance of the station from the distance to the destination. The *penalty-aware* utility function also tries to minimize unforeseen changes in prices. In contrast to the volatility-aware utility function, it penalizes required time to reach the gas station instead of rewarding closer distances. The *nearest station* utility function models user behavior that refuels at the closest gas station. Finally, the *random* utility function selects a gas station among the identified gas stations within the search radius randomly.

To evaluate the performance of the different utility functions, we aimed at defining a representative data set concerning different routes all over Germany and several dates during the year 2018. The 22 selected routes cover Germany's roadside to a large extent and show differing route lengths (between 50 km and 900 km) and used road types. The 18 dates are chosen with regards to vacations or holidays, regular weekdays, and weekends. For each date, three different timestamps are used as the start time of the journey: 6:00 AM, 12:00 PM, and 4:00 PM. The average cost per route is around 25 €. Our hypothesis of increasing uncertainty with a longer planning horizon implies that the longer the planning horizon, i.e., the longer the route, the larger the difference between estimated and actual costs. Therefore, we categorized the routes into three categories: (i) shorter than 100 km, (ii) between 100 km and 400 km, and

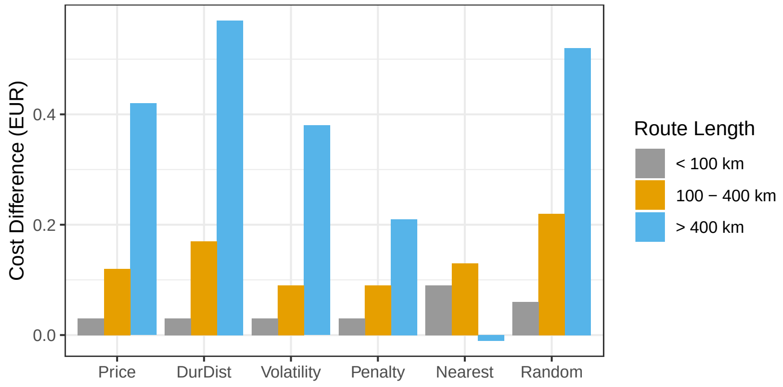


Figure 6.7: Difference between estimated and actual costs for each utility function with regards to three categories of route lengths. The colors represent the differing route lengths, and the bars show the cost difference in EUR per utility function.

(iii) longer than 400 km. We calculate the mean difference between estimated costs at planning time and the actual total costs for each category and utility function. Figure 6.7 visualizes the difference between estimated and actual costs for each utility function with regards to the three categories of route lengths. The gray bars ($<$ 100 km) show for all utility functions except the nearest station the smallest difference between estimated and actual costs of around 0.03 € and 0.09 €. The orange bars (100 - 400 km) show a noticeable increase in cost difference between 0.09 € and 0.22 €. The blue bars ($>$ 400 km) show especially for the duration-/distance and random utility function strong increases in cost differences between 0.52 € and 0.56 € while the cost increase for the other utility functions is less significant. The nearest station utility function does not reflect these characteristics as only a slight increase can be shown between short and medium route lengths, and is even reduced slightly for long routes. This can be explained by the selection criterion since the travel time to the gas station is short and price changes occur very rarely. These results indicate that the size of the planning horizon and that integrating uncertainty parameters into utility functions is necessary to handle unforeseen situations.

6.2 Logistics

Analogous to the ITS domain, the discussed aspects of adaptation planning can also be transferred to diverse other domains. According to the No-Free-Lunch theorem for optimizations [WM97], there is no single algorithm that performs best on all optimization problems and their cost functions. It is rather the case that if an algorithm works well for one problem, it will securely perform worse for other problems[1]. Hence, the situation and problem dependent behavior of adaptation planning systems can also be transferred to other domains. Further, fairness of the selected methods and determined actions as well as uncertainty in the environment need to be addressed in adaptation planning systems in general. To demonstrate the applicability of the mentioned aspects and the performance of our concepts, we selected logistics as second use case domain. In the following sections, we present two use cases from the logistics domain: (i) storage assignment and order picking, and (ii) vehicle routing. For both domains we introduce the basic problem statements, discuss how already mentioned adaptation planning aspects transfer to it, and highlight important challenges these use cases add to this work.

6.2.1 Storage Assignment and Order Picking

This section introduces the storage assignment and order picking problem as introduced in our publications [LMK$^+$21a, LMK$^+$21b]. Warehouses play a central role in the supply chain of a company and contribute to its logistical success. When employing humans, picker-to-parts and parts-to-picker methods are differentiated [DK07]. Experts estimate the picker-to-parts system to be the most common in Western Europe with a share of over 80% [DKLDR07]. A well-known picker-to-parts system is the mezzanine warehouse. Working within a mezzanine warehouse consists of two main tasks: (i) filling the storage with goods (storage assignment) and (ii) picking items out of the storage (order picking). The storage assignment problem defines the task of selecting storage locations to put a product into storage. Since mezzanine warehouses usually provide a large number of storage racks, it is difficult to find the optimum storage allocation that fulfills all custom constraints defined for the problem. The order picking problem defines the task of computing a pick route that collects the requested products of a customer order. Finding suitable storage allocations is important, as the allocation of products affects the travel distances during order picking. Due to the NP-hardness and the complexity of the storage assignment [RSCMT19] and order picking problem [ÇS19], efficient algorithms are required to find satisfying solutions within acceptable time.

In our paper [LMK$^+$21a], we analyzed existing literature in this domain and showed that many approaches exist for optimizing both warehouse problems. Further, each approach focuses on a specific detail of these warehouse problems

[1]We refer the reader to [WM97] for a mathematical definition of assumptions and theorems as well as for a mathematical proof.

and according to the No-Free-Lunch theorem for optimizations [WM97], there is no single algorithm that performs best on all problems. The theorem also states that the algorithm's observed behavior to date does not necessarily represent a good prediction for future behavior. Hence, the optimization technique needs to be selected based on the exact problem definition and its parameters need to be tuned to fit the current situation.

The second aspect of our work, that is fairness, can also be transferred to this domain. As already stated, mezzanine warehouses are picker-to-parts systems where human workers need to fill the storage with goods and pick items out of the storage. Order picking is known to be the most labor-intensive and costly task in which the employees account for a large part of the warehouse performance. The fairness of the work compared to other employees or other shifts is a central factor to achieve a good acceptance of the planning system, and employee engagement and performance [HRMS98, HRFA19]. Hence, an adaptation planning system needs to keep track of the fairness of determined actions that impact the work of employees.

Uncertainty as third aspect of this work can also be applied in the warehouse domain. Since a mezzanine warehouse is a highly dynamic warehouse in which a continuous flow of goods needs to be maintained, uncertainty exists in the stability of the current status of the warehouse. In particular, the planned storage location for one item is determined based on the already stored items. However, the next item to be stored might influence the goodness of the storage rack for the last items. Additionally, since the warehouse is operated by people, the work pace is not deterministic. Some employees might require more time to walk to a specific location or to find the determined rack position. Hence, some uncertainties occur in this problem statement and need to be taken into account while planning the next adaptations.

The most important research challenge this use case adds to this work is the inter-relatedness of the handled problem statement. In the literature, many approaches exist for optimizing both warehouse problems. However, most approaches usually target either of the warehouse problems; some works target both problems, however miss to integrate the interrelation between them and view each problem separately [vGRCdK18]. However, as identified by [GGM10], warehouse problems are strongly coupled. Thus, optimizing each warehouse problem individually may yield sub-optimal solutions, harming the overall warehouse performance. Hence, we postulate the need for an integrated approach that handles the storage assignment as well as the order picking problem. This approach should be aware of the impact of decisions in one problem on the other problem and act accordingly to address both problems simultaneously. This research challenge is related to research in the systems-of-systems domain [Mai98].

6.2.2 Vehicle Routing Problem

This section presents the rVRP as introduced in our publication [LKK$^+$21a] and our technical report [LKK$^+$21b]. In the last two decades, the demand for road freight transport increased worldwide; for example, in Germany it increased by 150 billion ton kilometers to around 500 billion ton kilometers [Kor21]. Developments as increased just-in-time production and online shopping will further increase those numbers in the next years. A key success factor for this trend is the efficient and correct planning of tours for transports which means to solve the VRP. The classical VRP specifies the assignment of customer orders to vehicles and the optimization of their tours [GRW08], which refers to solving the underlying TSP. In contrast, the rVRP includes additional constraints required for a real-world application, such as P&D behavior, TW, pause times, trailer capacities, and driver assignments. Tim Pigden stated that the original model of the VRP does not match real-world applications since it does not include concepts of order, separate resources corresponding to the driver, the tractor unit, and the trailer [Pig13]. Therefore, this model leads to inaccurate results when applied in practice and new approaches to address the various aspects of the rVRP are required. Since the rVRP is an NP-complete problem, exact solutions are hard to calculate in reasonable time and, hence, logistic companies often use meta-heuristics to find so-called good enough solutions in a reasonable time.

In our paper, we analyzed the existing literature capturing diverse problem variations of the VRP. Table 6.2 provides an excerpt of the literature on VRP such as the Capacitated Vehicle Routing Problem (C-VRP), Vehicle Routing Problem with Time Windows (VRP-TW), Vehicle Routing Problem with Pickup and Delivery (VRP-PD), and Vehicle Routing Problem with Time Windows and Pickup and Delivery (VRP-TW-PD). As can be easily seen, great interest in researching solutions for all kinds of the VRP. Similar to all previously discussed use cases, the No-Free-Lunch theorem for optimizations also applies to these problem statements and the selection of the best performing algorithm and its parameterization is situation and problem dependent.

The fairness aspect can also be transferred to the VRP domain. Once again, the human factor comes into play for this aspect. As the tours and routes computed by the optimization algorithms need to be driven by human beings, the fairness aspect is highly relevant. Within a company, no differences should be made when it comes to the physical stress during loading and unloading

Table 6.2: Excerpt of the related work covering diverse VRP problems classified by their algorithm type. (EA = Exact algorithm, SA = Swarm algorithm)

	C-VRP	VRP-TW	VRP-PD	VRP-TW-PD
EA	[FTV94, AHM13, CLSV07, FLL$^+$05, LN87, QTY10]	[HI20]	-	-
LS	[GC09, WZZL18, RSM20]	[CC02, Ski11, CM12]	[MJMBMD17, Aa06]	[LL02, WZW$^+$16]
SA	[SWH11, lCkYmW06, BM04, DX06, YYY09]	[BS03, RMLG07, FMP07]	[WS03, CHD07, CEE16, Çat09]	[DHR00, TYA$^+$17]
GA	[BA03, BB03, GLP02, VCG$^+$12, CM13]	[GG10, KTSA14, ENOBTMG16]	[PDG96, TG10, SR16]	[Pan05, WC12, CMMF17]

of a truck, but also to the required driving time. Similar to the balanced work within a warehouse, also the balanced workload for each driver is important for the engagement and performance of employees. Hence, the adaptation planning instance should be aware of the impact the adaptation decision has on the individuals within the system.

Since trucks are part of the road traffic, the planning of tours and routes is subject to great uncertainties in multiple dimensions. As already introduced in Section 6.1.3, fuel prices are highly volatile and the costs per route are therefore very dynamic. Furthermore, traffic on the road adds uncertainties in terms of travel time, as it is extremely dynamic due to road works, accidents and congestion. All these aspects of the VRP in the real world complicate the computation process of reasonable tours and routes and influence the performance of the whole company. Hence, these aspects should be integrated into an adaptation planning system and should be considered when planning the tours.

Similar to the most important challenge of the storage assignment and order picking use case, namely the system of systems optimization, this challenge also applies to the introduced VRP. Due to the high complexity of the problem, and inspired by [CT10], we divide the problem into two stages. First, we address the problem of distributing all orders, including pickup and delivery options, to the available vehicles (VRP-stage). In this step, several assignment-related constraints such as ordered quantity, weight, and size or forbidden co-located products are addressed. However, many of the constraints are sequence-dependent, such as driving, and service time, planned pause times or driven kilometers, and, hence, a nested TSP instance for each vehicle needs to be solved. The solved TSP instances are then sent back to the VRP-stage that performs an additional step to determine time windows and pause stops for each tour. In contrast to the warehouse example where both systems inside the overall system run in parallel, the systems within this problem statement are nested and the inner TSP instance needs to be solved to compute the final solution of the VRP. Hence, this use case presents another challenge in the systems-of-systems research direction.

6.3 Summary

In this chapter, we proposed our first main contribution, which focuses on **Goal A**: *Self-aware optimization of adaptation planning strategies with particular attention to the field of ITS and logistics*. Therefore, we answer **RQ A.1** by analyzing a set of use cases from the field of ITS and logistics and identifying important characteristics of adaptation planning strategies. Based on these findings, we design our component-based self-aware optimization framework and the optimization of systems-of-systems approaches in the following parts of the thesis.

Part II

Self-aware Optimization

Chapter 7

Self-aware Optimization Framework

Motivated by the No-Free-Lunch theorem for optimizations [WM97], which indicates that there is no single algorithm that performs best for all classes of problems, and the property study for the scenarios in the foundations, this thesis aims at designing and prototypical implementing of a generically applicable self-aware optimization framework. This framework optimizes the decision making of use cases with self-adaptation properties. It should receive information about the application and the adaptation planning strategy, such as the performance of the system, and should adjust the parameters of the adaptation planning strategy or replace the strategy with a more promising one. The framework should be generically applicable in a variety of use cases while reducing the manual effort as well as required expert and domain knowledge.

In addition, these general goals for the framework, Section 6.1.1, Section 6.1.2, and Section 6.1.3 motivated the meaningfulness of the framework by studying three properties of targeted systems: (i) Situation-dependency, (ii) Fairness, and (iii) Uncertainty. In our studies, we were able to show the situation dependence of SAS, with the conclusion that the choice of the adaptation planning strategy, but also of the optimizer to improve the strategy parameters, is not a *one fitting all* choice, especially in multi-objective scenarios. The second property concerns fairness which we define similarly to Rescher as "dividing goods or bads on the basis of general principles that pertain to everyone alike" [Res02, p.13]. In our case study of platooning coordination, we showed that it is important to ensure a fair distribution of benefits and disadvantages resulting from participation in a platoon. The third property is about uncertainty of a self-adaptive system operating in dynamic environments which we define similar to Ramirez, Johnson, and Cheng [RJC12, p.101]: "Uncertainty is a system state of incomplete or inconsistent knowledge such that it is not possible for a DAS to know which of two or more alternative environmental or system configurations hold at a specific point." Our uncertainty study is related to fuel price trends that affect the performance of proactive adaptation planning systems for route planning.

In our proposed framework, we directly incorporate the situation dependent property of adaptation planning systems and develop a component for situation-awareness. This component is responsible for determining the current situation of the system and its environment to identify the most promising strategy and parameter setting. Further, we propose to address fairness for the participating entities by designing a specific set of adaptation planning strategies that ensure a fair distribution of advantages and disadvantages between all entities. Finally, we propose to address uncertainty by proactively integrating an uncertainty measure into cost functions that are used to determine the most promising adaptation plan. This way, we propose to incorporate uncertainty measures already at design time to anticipate possible negative effects of the uncertainty.

In line with our property studies discussed in the foundation part, we also use the platooning coordination use case as representation for ITS and running example in this chapter. We prototypically apply our concept of a self-aware optimization framework on the platooning coordination strategies and highlight how the concepts can be transferred to any other meaningful use case. Then, we explain abstract concepts of the framework using examples in this use case and show how the framework can be applied to it to illustrate the effort required to transfer it to other use cases.

We begin this chapter by first defining a set of assumptions that we specify for the design of our framework in Section 7.1. Afterwards, we define the Terminology of this chapter in Section 7.2. In Section 7.3 we introduce our system model, for which we use the layer architecture proposed by Kramer and Magee [KM07]. Section 7.4 discusses the concept of the framework from the point of view of a control loop and adapts the LRA-M Loop introduced by Kounev *et al.* [KLB$^+$17]. Afterwards, Section 7.5 presents the architecture of the framework and provides an overview of the designed components and their interaction, consisting of a Coordination, the Domain Data Model (DDM), the Situation Detection, the Strategy Selection, and the Parameter Optimization components. Section 7.6 is the final section addressing the design and concept of the framework by introducing a use case-specific adapter for the framework used in our running example of platooning coordination. Afterwards, we present our fairness-ensuring adaptation planning strategies for platooning in Section 7.7 that indirectly compensate negative effects for platooning members by distributing them equally among all members. We address uncertainty in Section 7.8, which presents our methodology in proactive adaptation planning strategies. We conclude this chapter in Section 7.9 and answer the corresponding research questions for this chapter. The content of this chapter is based on our publications [Les20, LHKK21a, LHKK21c].

7.1 Assumptions

In this section, assumptions are made for the design of the framework to ensure broad applicability in various use cases. The following assumptions ensure the proper operation of the framework as well as the use case and define the interactions between both systems. At the same time, they point out limitations that can be addressed in future work.

First, we assume that the use case for which the framework is to be used consists of two parts. One part is the environment in which entities operate based on their individual goals and actions. The second part is an adaptation planning system that monitors the entities and decides upon adaptation actions based on global goals. We assume that the operating entities adhere to the given plan of the adaptation planning system and execute all given adaptations. If they cannot implement these instructions, they report this to the adaptation executor, who then decides on further actions that should be taken by the entities. Further, we assume that the communication between entities in the use case and adaptation planning system is flawless and that the entities regularly report measurement and observation values to the adaptation planning system. Additionally, we assume that the strategy of the adaptation planning system is interchangeable and has the possibility to change its parameters at runtime.

Second, we assume that the use case to which the framework is to be applied is digitized, meaning that performance and monitoring data are captured and stored digitally—typically centrally in the adaptation planning system. Further, the adaptation planning system is able to transmit relevant data to a defined management entity—in this work the framework—where higher level optimizations take place. We assume that the interaction between framework and adaptation planning system of the use case is always successful. Therefore, we exclude any case where the connection between the two systems fails or the computed changes cannot be transmitted to the adaptation planning system due to other failures - resilience management of both systems is part of the future work of this thesis.

Third, we assume that the adaptation planning system works independently of a higher-level optimization, i.e., the framework, and can be used with a previously defined strategy algorithm and parameter set. Thus, it remains functional regardless of whether the framework determines an optimization adjustment. This is especially important in the startup phase of the framework, when optimization adjustments have not yet been determined. We also assume that this adaptation planning algorithm works independently and flawlessly and does not need to be monitored for failures.

Finally, we assume that the framework provides optimized decisions to the adaptation planning system without explicit request. Furthermore, we assume that the adaptation planning system regularly retrieves and successfully implements these changes. We assume that the adaptation planning system reports its current configuration along with other monitoring data to the framework to execute the optimizations based on the current state at given time intervals. Based on this data, future work can extend the framework to include a mechanism to ensure successful implementation of new strategies and parameter settings in the adaptation planning system.

7.2 Terminology

In this section, we define the terminology used in the following chapters to avoid misunderstandings and imprecise expressions. We start with the definition of the use case as well as the entities and their actions in the use case, continue with the definition of an observation, a context, and a situation, and finally define the term framework.

Use Case: We define a use case as a group of entities operating in a particular environment, pursuing their own goals. Entities in the use case can be linked to an adaptation planning system that helps them achieve their goals more efficiently, or that adapts the entity's actions to achieve global, regional, or local goals. The complexity and abstraction level of a particular use case are irrelevant as long as a adaptation planning system is in place. This adaptation planning system must provide multiple adaptation planning strategies and can provide configuration options. With respect to the running example platooning coordination, the use case could be defined at the regional level, e.g., as coordination of platooning of vehicles on a road segment with a central PCS fulfilling the role of the adaptation planning system. The use case could also be defined at a lower level, such as optimizing the inner platoon structure, i.e., the order of vehicles within the platoon (cf. to Section 6.1.2 for a motivating example).

Entity in a Use Case: An entity within a use case is a machine, human, or other object that can receive and execute instructions from an adaptation planning system. Entities may have the ability to make decisions for themselves according to their individual goals and do not necessarily need to receive instructions from the adaptation planning system. Entities may also work with coarse-grained instructions or work toward individual goals. The entities within the use case are expected to follow the adaptation actions they receive from an adaptation planning system, even if their individual goals dictate a different direction. For a discussion of research challenges related to coordinating global, regional, and local goals, we refer the interested reader to our publication [LKT19b].

Actions of an Entity: Entities of a use case have a given set of possible actions that they can execute to accomplish certain tasks or achieve their individual goals. The actions to be taken can either be specified in fine-grained terms by an adaptation planning system, or entities can work autonomously toward a coarser-grained goal. The second case also means that entities can operate without an adaptation planning system if the entities' goals are defined and the available actions enable the entities to achieve that goal.

Observation: An observation contains information about the use case at a particular point in time. This includes details about the entities, their sensed data from the environment as well as the current configuration of the adaptation planning system and its performance. These performance indicators must be defined individually for each use case. Using expert knowledge, the performance of the adaptation planning system can be evaluated based on these indicators. We define each observation as a triple $(context, input, metrics)$ at a given point in time that is sent to the system on a regular basis. The *context* represents a set of values used to determine the current situation of the use case. The *input* parameters are the configuration parameters of the adaptation planning system. The *metrics* are a set of indicators that represent the current performance of the adaptation planning system.

Adaptation Planning System: An adaptation planning system is a mechanism that uses the retrieved observations from a use case in order to plan adaptations within a SAS. These systems aim to identify changes in the environment and the system itself and to react accordingly and apply adaptation planning strategies to plan adaptations. The strategies are exchangeable within a SAS and require the configuration of parameters. These properties can be used to tune the performance of SAS by an optimized selection of adaptation planning strategies and parameter tuning.

Situation: We define a situation as a set of observed contexts that have similar values. This means that environmental factors, entities, and entity behavior occur in a similar combination to previous context data. We use the term situation from a technical point of view, following Cámara's definition, "where a situation includes at least the elements of the situation [...], and environmental factors and their current states" [CBK+17, p. 38]. We use the context data to determine the situation the system is in. The knowledge of the current situation is then used to adjust the adaptation planning strategy and its parameters to optimize the overall performance of the system. We also use this information to learn good strategies for each situation and improve system performance for future situations.

Framework: We consider a framework as an abstract modular application that defines a specific process structure, pursues a specific goal, and provides generic functionality by combining components. We assume that these components have well-defined interfaces through which they can communicate with other components to ensure smooth integration into the overall framework. As part of this work, we have implemented all relevant components of our framework. In addition, we have designed the framework to provide the ability to extend it by adding new components or customizing existing components. Furthermore, the user of the framework has to define a configuration for the specific use case, which defines the composition, setup and configuration of the framework and the components used.

7.3 System Model

This section introduces the system model we use for defining the self-aware optimization framework. The system model is presented in Figure 7.1 and integrates three layers following the three layer architecture proposed by Kramer and Magee [KM07] to incorporate the principles of maintainability and separation of concerns: (i) Application, (ii) Adaptation Planning, and (iii) Self-Aware Optimization. In the following, we explain the details of each layer and introduce the core contribution of this chapter: the self-aware optimization framework.

We refer to the bottom layer ① of the system model as the application layer and consider real-world use cases from ITS and logistics, as discussed earlier in Chapter 6. Entities of the use case monitor themselves and their environment. The collected data is sent to the upper layer where the adaptation planning system receives the data. After an adaptation planning cycle, the use case entities can receive adaptation actions to follow and execute. If the entities fail to carry out these instructions, we assume that they will report this to the adaptation planning system, which will decide on further action.

The middle layer ②, called adaptation planning, includes the adaptation planning system, which receives observations from the use case. The adaptation

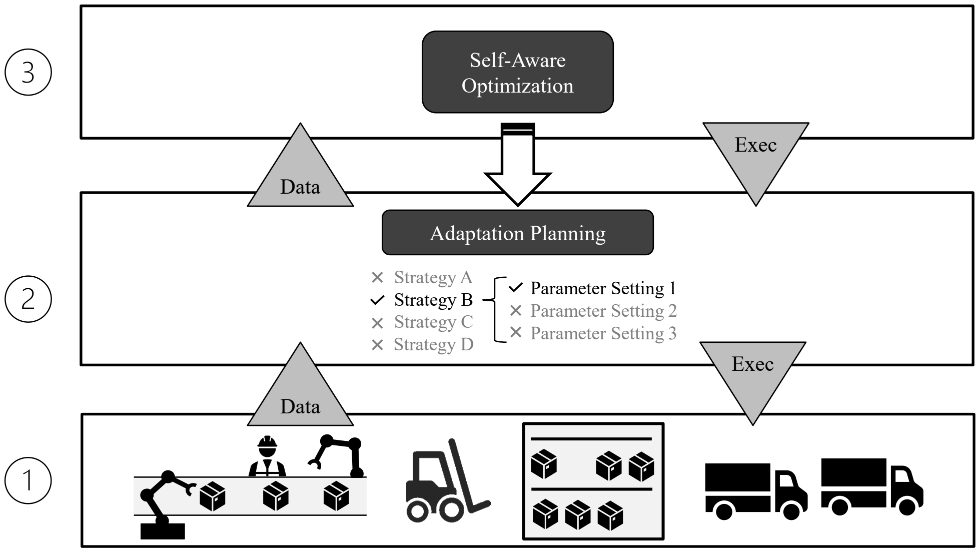


Figure 7.1: Multi-layer architecture of the self-aware optimization framework. Layer 1 represents an adaptive system, the adaptation planning system is shown in Layer 2, and Layer 3 shows the core contribution of this chapter, the self-aware optimization.

planning system applies a strategy that uses the received observations to plan adaptations for the managed system. These strategies are selected from various existing strategies in the adaptation planning system. The adaptation planning strongly depends on the use case and is therefore out of scope of this work The strategies can range from simple rule-based algorithms to complex (multi-objective) optimization algorithms. Furthermore, we assume that the user of the framework will provide multiple strategies per use case, customized for the particular use case, to provide the possibility of strategy exchange when needed. The performance data of the selected strategy is collected and—together with the use case's monitoring data—transferred to the next layer, which performs a self-aware optimization. After one self-aware optimization cycle, the adaptation planning layer may receive instructions to change the strategy parametrization or even to replace the strategy. As mentioned earlier, we assume that the adaptation planning layer executes these commands without interference.

Finally, the third layer ③ is called self-aware optimization. This layer is responsible for optimizing the parameters of the selected strategy in the adaptation planning layer as well as for the selection of strategies for the ② layer and, therefore, integrates several components: (i) situation detection, (ii) algorithm selection, and (iii) parameter optimization. The situation detection component receives the monitoring data, that is, the use case observations, and the performance data from the adaptation planning system and categorizes the observation into a currently present situation. The algorithm selection component uses the information about the current situation, combines it with experience from similar situations in the past and selects the most appropriate adaptation planning strategy. Finally, the parameter optimization component also receives monitoring data and tunes the parameters of the adaptation planning strategy. All decisions—including the situation, the selected strategy, and the parameter settings—are used in combination with monitoring and performance data to learn from previous decisions. A knowledge base manages the set of known situations as well as corresponding decisions and continuously learns which parameter and algorithm combination fits best for the situations already experienced. In addition, it is possible to develop another component that includes prediction and forecasting mechanisms to enable proactive adaptation of the system. Finally, the third layer passes the decisions to the adaptation planning layer ②, which executes them.

7.4 LRA-M Loop Adoption

In this section, we present our concept of a self-aware optimization framework from the control loop point of view. This perspective allows us to elaborate on the idea of the framework and explain the interplay between ongoing learning and reasoning in the framework. Since we use the terminology of self-awareness in this work, we focus this section on the corresponding LRA-M Loop. The LRA-M Loop was first introduced by Kounev *et al.* in 2017 [KLB+17] in his work on Self-aware Computing Systems. This loop is quite similar to other concepts like the MAPE-K Control Loop [KC03] or the Observer/Controller concept [TPB+11] and most of these concepts can be transformed into each other [LKT19b]. However, the LRA-M Loop explicitly includes a *Learn* and a *Reason* component. Learning allows the system to learn models of the system itself and the environment, while reasoning uses these models to trigger adaptation actions that modify the system and affect the environment. These components are essential parts of the framework because learning enables the framework to form models of the environment, i.e., the two lower levels of the system model (c.f. Section 7.3) and to recognize new situations. Reasoning gives the framework the ability to consider which adaptation actions might be beneficial in a given situation based on the knowledge of recent decisions or decisions in similar situations. This combination of ongoing learning and model-based reasoning forms the basis for the proposed framework, which

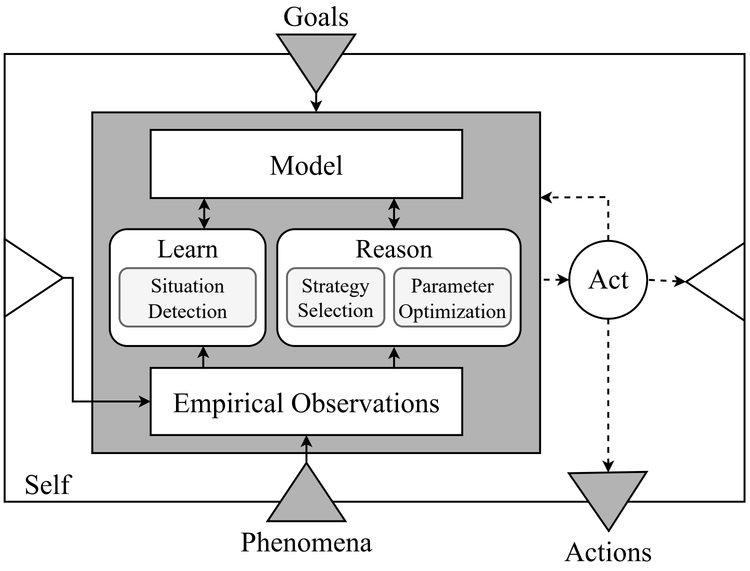


Figure 7.2: Modified Learn-Reason-Act-Model Loop (LRA-M Loop) based on Kounev *et al.* 2017. The basic LRA-M Loop is extended to include analysis and the meta-optimization in the Learn module and planning through optimization in the Reason module.

is why we chose to use the LRA-M Loop. Since the LRA-M Loop is a general-purpose concept applicable to diverse systems, we modify the control loop to explicitly include the functionalities of our framework, as shown in Figure 7.2.

The loop displays the system, also called the self, and its interfaces with the environment. It interacts with the environment by (i) perceiving *Phenomena* and storing them as *Empirical Observations*, (ii) receiving *Goals* to be achieved, and (iii) executing *Actions* based on the decisions made. The Empirical Observations are captured in the use case, i.e., the application layer of the system model, and used in the *Learn* and *Reason* modules. Furthermore, the decisions of the adaptation planning layer are part of the captured phenomena since they are needed as additional sources of information for the third layer. In the ongoing learning process, the observations are abstracted into models that contain knowledge about the environment and the system itself. We add the *Situation Detection* component into the Learn module, which enables to interpret the observations and updates the models to persist all gathered information. So far, we use clustering algorithms in the Situation Detection component to determine the current situation. However we have built each component in a modular fashion so that it is easy to extend the techniques used. Further, the learning component receives performance data of the managed use case with periodic observations and learns the impacts of the actions taken based on the current situation. This enables the system to continuously improve its reasoning and acting, and to keep the system's models of itself and the environment up-to-date. These models serve as the basis for the reasoning process that determines actions to be taken in response to a changing environment. The reason module determines actions for the adaptation planning system to adapt to changes in the environment or to deteriorated performance values. Hence, we assign the two components (i) *Strategy Selection*, and (ii) *Parameter Optimization* to this module. The Strategy Selection component combines the information from Situation Detection, the current use case performance with the learned models about the use case and determines whether to keep the current strategy or switch to another existing strategy. The Parameter Optimization component applies optimization techniques to tune the parameters for the selected strategy. So far, we use known, well-performing parameter settings as initial values for the optimization process to achieve a faster convergence of the optimization. The Situation Detection, Strategy Selection, and Parameter Optimizations in the modified LRA-M Loop are newly introduced components and not part of the original definition of the LRA-M Loop. These three components build the main contribution in terms of the proposed framework and are meant to be generically applicable to a wide range of suitable use cases.

7.5 Framework Composition

This section presents the composition of the generically applicable self-aware optimization framework. The framework consists of several components that configure the framework, store its observations, and execute the desired functionality, that is, to determine which strategy algorithm and parameters to use in the adaptation planning system. Figure 7.3 provides a comprehensive overview of the framework's structure. In the following, we briefly introduce each component and state its main contribution to the framework. All details of the components can be found in the following sections.

First of all, the user of the framework can use the *Domain Data Model* (*DDM*) to configure the entire framework and all its components. The DDM is the only part of the framework that the user needs to configure with use case specific information. Therefore, the framework considers the two lower layers from Figure 7.1 as a black box, of which it only knows the information defined in the DDM. In the DDM, relevant information about the use case such as the name and existing strategies in the adaptation planning system are defined.

The context part of the DDM defines what sensor data the adaptation planning system sends to the framework with respect to the context of the system. With regards to the platooning coordination use case, this sensor data could be the number of cars and trucks on the road, the platooning percentage, or the average speed of the vehicles. The parameter options of the DDM specify which configuration parameters exist for the strategy of the second layer and which values they can accept. Finally, the DDM contains a definition of metrics used to assess the performance of the use case.

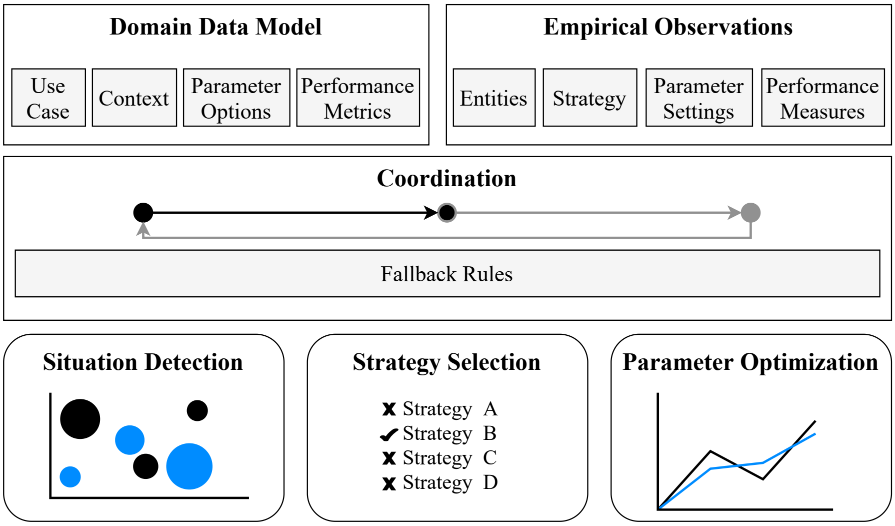


Figure 7.3: Composition of the self-aware optimization framework. The framework contains the Domain Data Model (DDM) for configuration, the Empirical Observations as a repository, a Coordination component that manages the workflow, and the three main components Situation Detection, Strategy Selection, and Parameter Optimization.

The second component of the framework manages all sensor data received from the use case and is called *Empirical Observations*. This component processes incoming data and provides an interface for the other components to retrieve the relevant data for the according computation step of the framework. For example, it maintains information about the entities of the framework such as the number of vehicles, the platooning percentage, and the vehicle speed. It also obtains information about the currently executed adaptation planning strategy, its parameter settings, and performance metrics. This enables the framework to reflect on previous adaptation decisions and learn which combination of strategy and parameter settings works best in a given situation.

The central component of the framework is the *Coordination*, which is responsible for retrieving the required data from the observation storage and passing them to the next component whose execution it triggers. This component is constantly active and regularly invokes the other components of the framework, namely the *Situation Detection*, the *Strategy Selection*, and the *Parameter Optimization*, in this predefined order. In the event that one of the other components fails, the coordination component can fall back to user-defined fallback rules from the DDM to remain functional. These fallback rules can be simple if-then-else rules, but since we provide the possibility to load arbitrary Python code into the fallback rules, the user could also extend the framework with a more sophisticated fallback mechanism.

The *Situation Detection* component of the framework receives the observation data of the use case, such as the entities and their current state, and determines the situation the use case is currently in. So far, we only use clustering algorithms for this purpose. However, it is easy to extend the component with other approaches, as we have designed the framework to be modular and the approach used is configured in the DDM. The identified situation is then returned to the Coordination component, which forwards this information to the Empirical Observations component.

After the Situation Detection completes its computation, the Coordination invokes the *Strategy Selection* component. This component combines knowledge about the current situation with experience from previous decisions in similar situations and determines which adaptation planning strategy is most appropriate for this situation. This decision is returned to the Coordination component that triggers the next component.

The last component of the framework is the *Parameter Optimization* component. This component receives the current parameter settings as starting point, historical data for the current situation, the corresponding adaptation planning algorithm, and performance measures. It then performs an optimization process to tune the parameter setting for this adaptation planning strategy to the current situation. It then returns the settings to the Coordination component, which stores all the collected information of this round of execution from the components, updates the system models, and sends adaptation actions to the adaptation planning system in layer two.

In addition to the general composition of the framework, we illustrate the workflow of the framework as a sequence diagram in Figure 7.4. The user is shown on the left side of the sequence diagram. He configures and starts the framework using the DDM, sets up the use case and configures it. The use case then starts its execution and sends the defined observations to the

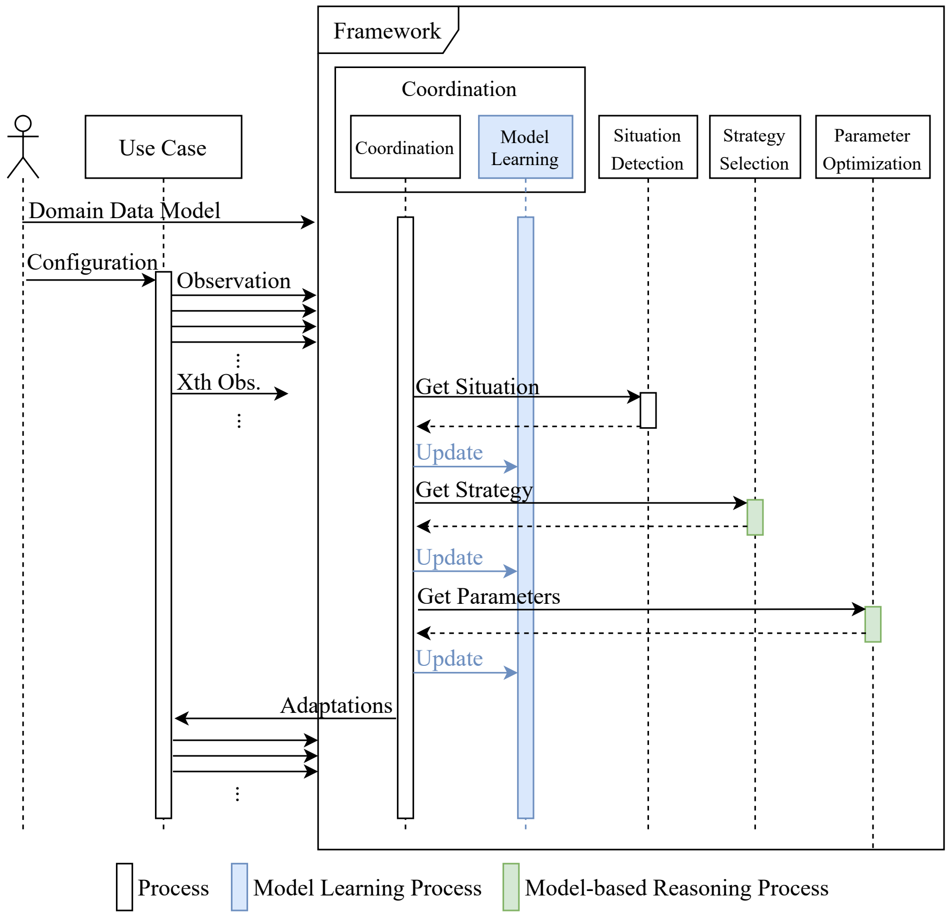


Figure 7.4: Sequence diagram of the workflow of the self-aware optimization framework. The user configures the framework and the use case sends observations. The framework processes the observations, identifies the current situation, selects the strategy and parameter setting, and continuously learns and updates its models.

framework in regular intervals, regardless of the current computational state of the framework. The Coordination component of the framework processes the incoming observations and forwards them to the Empirical Observations. After a certain number of received observations, the Controller component triggers the first execution of the Situation Detection component and forwards relevant observation data to this component. In the meantime, the Coordination component receives further observations from the use case, which are stored but not used until the next round of execution. After the situation is detected, this component returns the situation ID to the Coordination, which updates the system model of the environment. Then, the Coordination component triggers the Strategy Selection component with filtered observation data containing only observations of the identified situation. This component applies model-based reasoning based on this data to determine the most promising adaptation planning strategy. Again, this decision is fed back to the Coordination component which again updates the system model. Finally, the observed data is filtered again to include only data for the current situation and the adaptation planning strategy determined by the Strategy Selection. With this data, the Coordination triggers the model-based reasoning of the Parameter Optimization, which performs an optimization process to find the best parameter setting for the current situation and the selected strategy. After the Coordination component obtains this parameter setting, it updates the system model and sends adaptation tasks to the adaptation planning system, which executes them. This step completes one round of execution in the framework and after a predefined waiting time, the Coordination starts the next round.

7.5.1 Coordination

This section provides a more technical view of the Coordination component depicted in Figure 7.3 and extends the descriptions of the previous sections. We further summarize the workflow of the Coordination component using Pseudocode in Algorithm 1. The Coordination is responsible for initializing and invoking all other components of the framework. It also processes incoming observations and updates the system models based on observations and the framework's adaptation decisions. It is triggered at the start of the framework and instantiates all components of the framework (lines 1-2). To do so, the Coordination receives the DDM specified by the user, in which he defines the configuration of all components. It parses the DDM and instantiates the other components which completes the setup process of the framework.

The use case that the framework is intended to optimize is responsible for sending observations on a regular basis. Each observation consists of the use case entities, the currently active adaptation planning strategy, its parameter settings, and the use case performance metrics. Each new observation received triggers a new round of execution in the Coordination component. As a first step, the component uses the received data to compute additional important information relevant to subsequent processing (line 3): the time that the currently active parameter setting was active and the HV of the use case performance metrics. We require a user-defined waiting time in the DDM to allow adjustments to take effect. Thus, the framework calculates the time that the current

Algorithm 1: Pseudocode workflow of the Coordination component.

```
Input: DDM, new observation, existing observations
1  if start of framework then
2      initialize components defined in the DDM
3  derive additional information from the observation
4  save new observation
5  situation ← invoke Situation Detection on all observations
6  if situation could not be determined then
7      adaptations ← apply fallback rules to all observations
8      update system model with current adaptation decision
9      send adaptations
10 else
11     update system model with current situation
12     if waiting time after previous adaptation action is over then
13         if same situation as before AND number of optimization attempts not
             met then
14             parameter setting ← invoke Parameter Optimization on
                 observations of current situation and strategy
15         else
16             strategy ← invoke Strategy Selection on observations of current
                 situation
17             parameter setting ← invoke Parameter Optimization on
                 observations of current situation and strategy
18         update system model with current adaptation decision
19         send adaptation decision to use case
```

configuration is active in the use case and waits a predefined amount of time before evaluating the performance of the latest adaptation decisions to reduce unstable effects after recent changes. This also prevents too many adaptation actions from being sent to the use case without enough time for implementation. The HV measure is a widely used quality indicator for multi-objective optimization, especially in evolutionary optimization (c.f. Section 3.3). We use the HV to reduce the observed performance indicators of the use case to a single performance value. This allows us to use any single-objective optimization technique in the Parameter Optimization component without requiring multi-objectiveness for this technique. Afterwards, the component forwards the observation and derived information to the Empirical Observations component (c.f. Figure 7.3) that stores the incoming data (line 4).

Then, the Coordination passes the new observation to the Situation Detection component (line 5). Since the Situation Detection component applies clustering algorithms for identifying the current situation, it needs all the observation data collected from the use case for each execution. Therefore, we decided to implement an additional internal data management for the Situation Detection component to reduce the communication and data transfer between the components. After the Situation Detection identified the current situation, it returns the situation to the Coordination. If the available observation data is not sufficient for the clustering algorithm or the current situation is clustered as noise, the Situation Detection does not return a situation.

The Coordination component checks whether the situation detection was successful and returned a situation (line 6). If the situation detection did not return a situation due to insufficient data or classification as noise, the Coordination component applies the fallback rules to the current observations (line 7). Then, the Coordination updates the system model with the most recent adaptation decision and sends the adaptations to the use case (lines 8-9). In case the Situation Detection returned a valid situation (line 10), the Coordination adds information about the current situation to the system model. Since we apply a clustering algorithm in the Situation Detection that always clusters all observation data, it could restructure the whole data and find different clusters compared to the clustering of previous executions. In this case, the Coordination updates the system model and reclassifies the already clustered observation data to match the latest clustering (line 11).

After successfully updating the system model with respect to the current situation, the Coordination checks whether the waiting time after a previous adaptation action has expired (line 12). This waiting time is defined by the user in the DDM and serves as a cool-down period for use case adaptations to take effect. By doing this, we ensure that the transient phase of the use case is waited for and performance measures are retrieved that evaluate only the most recent adaptations. If the waiting time is still active, the current round of execution has ended and the Coordination waits for the next observations of the use case. When the waiting time has expired, new adaptation decisions can be send to the use case. In the next step, the Coordination requires another user-defined parameter from the DDM: the number of optimization attempts for the Parameter Optimization. This parameter specifies how many optimization cycles are performed per situation before a change in strategy is considered. This definition of optimization attempts per situation provides sufficient time to tune the parameters and avoids a hasty change of the selected strategy. The Coordination first checks if the current situation is the same as in the previous execution. Then, based on the user-defined parameter, it checks whether the necessary number of optimization attempts for this situation has already been executed (line 13). If this is the case, the Coordination requests all observations of the current situation and strategy combination and passes them to the Parameter Optimization. The Parameter Optimization computes a new set of parameters and returns it to the Coordination (line 14). However, if the number of optimization attempts has been exceeded this indicates poor performance of the currently used strategy which the framework uses to search for a new, better fitting strategy. In this case, or whenever the situation changed (line 15), the Coordination requests all observations of the current situation and passes them to the Strategy Selection component (line 16). This component uses this information to reason about the most promising strategy for adaptation planning. After the computation, this component returns the selected strategy to the Coordination. Then, the Coordination requests all observations of the current situation and the selected strategy to pass them to the Parameter Optimization (line 17). Using this information, this component performs an optimization task to select the most promising parameter settings for this strategy and returns the results to the Coordination. The Coordination, in turn, uses the strategy decision and its parameterization to update the system model (line 18). Finally, it sends the adaptation decisions including the strategy and the parameter setting to the use case (line 19).

To better understand the timing within the framework, we present an example timescale for invoking the three components Situation Detection, Strategy Selection, and Parameter Optimization in Figure 7.5. All timing values can be

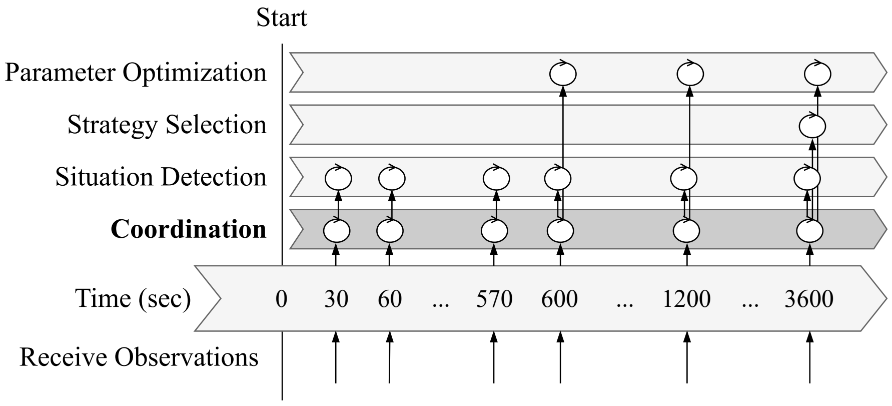


Figure 7.5: Timescale of the components and their computations the Coordination invokes. Observations arrive every 30 seconds and triggers an execution of the Coordination which then decides which other components to invoke.

defined by the user with respect to the use case. Therefore, the timing presented here should only be considered as an example for demonstration and not as the fixed timing of the framework for all use cases. For simplicity, we assume that no situation changes occur in this example. The figure shows time in seconds along the x-axis as a time scale, arranges the components above the time scale, and received observations are shown as arrows pointing to a specific time on the time scale. The use case in this example is configured to send current observations at a regular interval of 30 seconds. Each incoming observation triggers the Coordination that decides which other components are required at that time. At the beginning of the framework execution, the Coordination stores the received observations and forwards them to the Situation Detection. However, since there is not enough data at the beginning of the execution, the Situation Detection does not provide a situation and the Coordination applies the fallback rules. Once there is enough data (at second 600), the Coordination component triggers the Situation Detection that returns a specific situation ID. The situation identification then triggers the Parameter Optimization for the first time. Strategy Selection is omitted at this point because we decided to first optimize the parameters of the current strategy to see if the performance of the strategy can be sufficiently improved by an optimized parameter setting. Therefore, the user defines a number of optimization attempts that must be computed before the Strategy Selection can be triggered. This parameter is situation dependent and the number of optimization attempts is executed as long as the situation remains the same. If the Situation Detection component identifies a different situation than the last one, the Coordination triggers the Strategy Selection and Parameter Optimization regardless of whether the required number of optimization attempts is reached, which is set to five for the presented example. Thus, after 3600 seconds execution time, the Coordination has triggered five optimization attempts and triggers the Strategy Selection.

7.5.2 Domain Data Model

The DDM is a representation of the use case for the framework and serves as configuration file for the framework as depicted in Figure 7.3. It contains all use case-specific information to optimize the use case and thus enables the generic applicability of the framework for a variety of use cases. This means that these settings strongly depend on the chosen use case and can be enriched by use case specific parameters. Further, the DDM provides configuration information for the components that the Coordination component uses to instantiate the components. The DDM is defined using YAML Ain't Markup Language (YAML) and comprises four main parts: (i) use case, (ii) context,

(iii) parameter_options, and (iv) performance_measures. In the following, we describe each of these parts separately and provide a short example YAML file for this part. Since these examples cannot reflect every configuration option of the DDM, we provide the full specification in Section A.1.

We name the first part of the DDM *use case* (Listing 7.1, line 1) which contains general information about the use case. The *name* (Listing 7.1, line 2) of the use case is the first key of this part, which is used to identify all information collected during the execution. The second key is called *available_strategies* (Listing 7.1, line 3) and consists of a list of available adaptation planning strategies in the use case. The Strategy Selection component of the framework uses this list to determine the most promising strategy for the current situation. The framework refers to them as black-box strategies and sends the name to the second layer which is able to select the appropriate strategy identified by its name. This list of possible strategies does not need to be exhaustive and the user can omit strategies he does not want to be executed. The last key of this part is the *fallback_rules* key (Listing 7.1, line 4), which defines a path to a Python file that contains fallback rules for the framework. These fallback rules should reflect expert knowledge from the use case and are used by the framework in case the situation detection is not possible due to insufficient data or the current situation is identified as noise. Listing 7.1 presents the first part of the YAML file of an example use case called `platooning_coordination`. In this use case, two adaptation planning strategies `s_1` and `s_2` are available. Finally, the path to the predefined fallback rules is defined as `Path.To.Rules`.

Listing 7.1: Example for the use case part of the DDM YAML.

```
1  use_case:
2    name: platooning_coordination
3    available_strategies: ["s_1", "s_2"]
4    fallback_rules: "Path.To.Rules"
```

The second part of the DDM is called *context* (Listing 7.2, line 5) and specifies what *data* (Listing 7.2, line 6), i.e., observations, the use case sends to the framework. Furthermore, it defines the configuration of the Situation Detection component with the key *situation_detection_settings* (Listing 7.2, line 13). The data key of this part contains any number of context parameters from the use case, which can be named arbitrarily, but must be unique (Listing 7.2, lines 9-11). The framework will use these keys as identifiers when logging information to a database. Further, each context parameter requires a *data_type* specification (Listing 7.2, line 10,12) defined using `int` and `double` values.

Listing 7.2: Context part of the YAML definition of the DDM.

```
 5 context:
 6   data:
 7     # any number of context parameters
 8     # with unique names
 9     context1:
10       data_type: int
11     context2:
12       data_type: double
13   situation_detection_settings:
14     # available algorithms: RuleBased, kMeans,
15     # DBSCAN, OPTICS
16     algorithm: "DBSCAN"
17     settings:
18       min_samples: 120
19       eps: 34
```

The situation_detection_settings key describes the configuration of the Situation Detection component and consists of the two keys *algorithm* and *settings* (Listing 7.2, lines 16-17). The algorithm key expects the definition of an available situation detection algorithm. So far, four algorithms are available which we describe in more detail in the next section: `RuleBased`, `K-Means`, `Density-Based Spatial Clustering of Applications with Noise (DBSCAN)`, and `Ordering Points To Identify the Clustering Structure (OPTICS)`. We include the clustering technique K-Means in this set as it is the most common technique in machine learning. However, it requires the definition of k representing the number of clusters to be identified which increases the required domain knowledge and configuration overhead. This led to the idea to use density-based algorithms DBSCAN and OPTICS which do not require a predefined number of clusters but cluster the observations based on their density. Hence, they are able to identify any meaningful amount of situations observed from the use case environment. We selected DBSCAN as it is a commonly used density-based clustering technique. However, it also requires a parametrization which increases the required domain knowledge. The decision to use OPTICS addressed the requirement of reduced domain knowledge best as it operates with the least configuration overhead. Still, this set of clustering techniques should be considered as prototypical implementation and this list can easily be extended whenever another algorithm might perform better. Each algorithm requires additional configuration parameters that are part of the settings key. Listing 7.2 provides a short YAML example for the context part. It defines two context parameters `context1` and `context2` for the data key with data_type `int` and `double`. For the situation_detection_settings it is specified that the algorithm `DBSCAN` should be used and the required settings for this algorithm `min_samples = 120` and `eps = 34` are defined (Listing 7.2, lines 18-19).

The third part of the DDM is called *parameter_options* (Section 7.5.4, line 20). It defines input parameters of the adaptation planning strategy that can be tuned by the framework and provides configuration information for the Strategy Selection component. This part consists of the *options* for the input parameters and the *strategy_selection_settings* (Section 7.5.4, lines 21-34). The options key contains an arbitrary number of input parameter options for strategies and the key is in turn used as identifier for this parameter (Section 7.5.4, lines 24-28). Thus, it can be named arbitrarily but must be unique within this DDM. Each input parameter option further consists of three mandatory keys: *data_type*, *min*, and *max* and an optional key *strategies*. The *data_type* key defines the data type of the input parameter option, where we accept `int` and `double` (Section 7.5.4, lines 25,29). The *min* and *max* keys allow the user to specify the value range the input parameter can take (Section 7.5.4, lines 26,27,30,31). Finally, the strategies key allows the user to define for which adaptation planning strategy this input parameter is meaningful by defining a list of strategies (Section 7.5.4, line 33). This key is optional and the absence of this key leads to the conclusion that this parameter applies to all strategies. The second key of this part is the *strategy_selection_settings* key, which configures the Strategy Selection component. This key consists of five mandatory keys: *observations_between_adaptations*, *min_optimization_attempts*, *window_size*, *threshold_exceeds*, and *method* and one optional key called *hypervolume_threshold*. The key *observations_between_adaptations* defines the number of observations the framework must receive before new adaptation actions can be performed (Section 7.5.4, line 35). This property allows the user to define the transient phase for the use case where measurement data might be unreliable due to recent changes in the system. The *min_optimization_attempts* key defines the number of parameter optimization attempts for a situation before the Coordination component considers computing a new adaptation planning strategy (Section 7.5.4, line 36). The *window_size* and *threshold_exceeds* keys determine whether a new strategy should be chosen (Section 7.5.4, lines 37-38). For a detailed explanation of these keys, please refer to Section 7.5.4. Listing 7.3 provides a short YAML example for the parameter_options part of the DDM. It defines two options `param1` and `param2`, where the first one is of type `int`, accepts values in $[0, 100]$, and applies to all strategies. The second parameter option is of type `double`, accepts values in $[0.0, 2.0]$, and is applicable for the strategy `s_1`. Furthermore, it specifies the minimum number of optimization attempts to five.

Listing 7.3: Parameter options part of the YAML definition of the DDM.

```
parameter_options:
  options:
    # any number of context parameters
    # with unique names
    param1:
      data_type: int
      min: 0
      max: 100
    param2:
      data_type: double
      min: 0.0
      max: 2.0
      # optional definition of relevant strategies
      strategies: ["s_1"]
  strategy_selection_settings:
    observations_between_adaptations: 1
    min_optimization_attempts: 5
    window_size: 5
    threshold_exceeds: 3
    # available methods: hypervolume, threshold
    method: "hypervolume"
    hypervolume_threshold: 3.4
```

The last part of the DDM is called *performance_measures* (Listing 7.4, line 42) and defines indicators of the performance of the defined use case. This part contains any number of performance measures from the use case, which can be named arbitrarily (Listing 7.4, lines 43,47). Since these names are used as identifiers in the framework, they need to be unique. Each performance measure consists of three mandatory keys *data_type*, *higher_is_better*, and *reference_value*, and an optional key called *threshold_value*. The *data_type* specifies the performance measurement data type, which can be either `int` or `double` (Listing 7.4, lines 44,48). The *higher_is_better* key defines whether a higher or a lower value of this metric is better for this use case, and is of type Boolean (Listing 7.4, lines 45,49). The *reference_value* key specifies a reference value for the calculation of the HV, which needs to be of the same type as specified in *data_type* (Listing 7.4, lines 46,50). Finally, the *threshold_value* key is only required if the `threshold` method is selected in the *strategy_selection_settings* of the *parameter_options* part and defines a threshold value that cause the Strategy Selection component to compute a new strategy. Listing 7.4 provides a YAML example for the *performance_measures* part of the DDM and defines two performance measures `pm1` and `pm2`. The first is of type `int`, where higher values represent a better use case performance and a reference value of `-1`. The second performance measure is of type `double`, with lower values representing better use case performance and a reference value of `100.0`.

Listing 7.4: Performance measures part of the YAML definition of the DDM.

```yaml
performance_measures:
  pm1:
    data_type: int
    higher_is_better: True
    reference_value: -1
  pm2:
    data_type: double
    higher_is_better: False
    reference_value: 100.0
```

7.5.3 Situation Detection

The Situation Detection component is responsible for identifying the current situation the use case is currently experiencing as depicted in Figure 7.3. The use case periodically sends observation data to the framework, as defined in the context part of the DDM. The frameworks' Coordination component forwards this data to the Situation detection. So far, this component provides four methods for detecting the current situation: (i) rule-based, (ii) K-Means, (iii) DBSCAN, and (iv) OPTICS. As already mentioned, we provide this set of clustering techniques including K-Means as the most basic clustering technique requiring the definition of a number of cluster to be identified. Further, we select the density-based clustering techniques DBSCAN and OPTICS to reduce this requirement. While DBSCAN still requires manual configuration effort, OPTICS fits best to the requirement of the reduction of domain and expert knowledge. Still, this set of clustering techniques should be considered as prototypical implementation and this list can easily be extended whenever another algorithm might perform better. All methods operate on all context data available in the system. To reduce the communication overhead within the framework, the Situation Detection contains a duplicated set of received observation data within the component, and the Coordination only needs to forward the current observation. The Situation Detection component computes the current situation and returns a situation ID to the Coordination component. This ID is further used in the Strategy Selection and Parameter Optimization components to find appropriate adaptation decisions for this specific situation and to learn from previous decisions in this situation.

The situation detection process can be defined as a mathematical function that maps observation data from the use case to an integer value. This value represents the situation ID as defined in Equation (7.1). We define the value interval of this function as $[-1, \infty)$, where the value -1 indicates that the situation could not be detected. This could be the case for two reasons: First, the amount of available data is insufficient to determine the situation. Second, the observation data is classified as noise, meaning that the currently observed values cannot be classified as a specific situation. This could be due to a novel situation for which these is not enough data, or measurement inaccuracies in the use case. In the case that the Situation Detection classified the current situation as -1, the framework does not invoke any further computational processes, such as Strategy Selection or the Parameter Optimization. However, the Coordination component uses the user-defined fallback rules from the DDM (Listing 7.1, line 4) to determine any adaptation actions that may be required. If the returned situation ID is equal to or greater than zero, the Situation Detection component has determined a valid situation. Therefore, the Coordination component can invoke the Strategy Selection and Parameter Optimization components. The actual value of the situation ID does not allow for further interpretation regarding the similarity of situations. For example, if the component identified three situations $s_1 = 0, s_2 = 1, s_3 = 10$, it means that these three situations exist and are all different from each other. Moreover, the proximity of the values 0 and 1 does not mean that the situations s_1 and s_2 are more similar to each other than the situation s_3.

$$\mathit{sit\_det}(\mathit{context}) = \begin{cases} -1, & \textit{if} \text{ situation is classified as noise} \\ >= 0, & \text{otherwise} \end{cases} \tag{7.1}$$

Due to the ongoing nature of the framework, the use case regularly sends new observation data. Therefore, the amount of observation data grows as the framework is executed and the Situation Detection component receives more and more data to improve decision making. However, this could lead to a changed in the assignment of context data to situations during the execution time. This means, the situations identified during the last Situation Detection process may not be the same as those identified in the current process. Completely new situations or a change in assignment from an already assigned observation could lead to inconsistencies in the data. For example, a context observation classified as situation s_1 in the last process could now be classified as s_2 when more data is available. Therefore, the Situation Detection component updates its learned models after each execution to match the latest findings to the observation data.

We provide two types of situation detection mechanisms, one rule-based mechanism and four clustering algorithms that can be selected and configured by the user in the DDM. Since we designed the framework to be modular, it is easy to extend the framework with additional components or to further develop individual components with additional techniques. The following Algorithm 2 summarizes the workflow behavior of the Situation Detection component. The component receives the DDM and the new observation and selects the configured algorithm for the Situation Detection. In all cases, the component retrieves required parameters for the selected technique from the DDM and invokes the configured technique. All techniques return the `situationIDs` for all observations, that is, the cluster to which each observation in the data set is assigned. The component then update its situation model of all observed data with the latest classification and returns the `situationID` of the new observation to the Coordination component.

Algorithm 2: Pseudocode workflow of the Situation Detection component.

```
Input: DDM, new observation
1  switch DDM.situation_detection_settings.algorithm do
2      case RuleBased do
3          retrieve path to fallback rules from DDM
4          situationID ← execute fallback rules
5      case kMeans do
6          retrieve K-Means parameters from DDM
7          situationID ← invoke K-Means
8      case DBSCAN do
9          retrieve DBSCAN parameters from DDM
10         situationID ← invoke DBSCAN
11     case OPTICS do
12         retrieve OPTICS parameters from DDM
13         situationID ← invoke OPTICS
14 update situation model with latest classifications
15 return situationID of new observation
```

The rule-based situation detection offers the possibility to integrate domain knowledge in the identification process of this component. For example, in the platooning use case, the user could specify frequent traffic volumes for which he knows the best performing configuration of the adaptation planning system. The user defines the rules in form of a Python file that is loaded and executed by the component. As the simplest option, the user can define Event-Condition-Action rules to specify known, well-performing configurations. However, since the user describes the fallback rules in a Python file, he can also construct arbitrarily complex functions to identify situations. Still, the user must provide a script that matches our definition of the situation detection function in Equation (7.1). The user can adapt these rules for future executions of the framework as he gains new domain knowledge from running the framework and analyzing its decisions. In the context of this thesis, we omit updating the user-provided rule set with new knowledge from previous executions of the Situation Detection. This also results in the framework being unable to react to new situations in the fallback case, since they are not present in the rules. In this case, the new situation must be classified as noise. However, there are several approaches to automatically update rule sets during execution [NZJT12, CHS+18, GMS+09].

In addition to the static rule-based situation detection, we provide three clustering-based situation detection methods. Advantage of these methods are that they can automatically detect new situations due to their unsupervised learning approach, and that they do not require domain knowledge [ATL14, FGKV19]. One clustering algorithm we integrate into our framework is K-Means in two versions. The first version works with a predefined parameter k that specifies the number of clusters to identify. In the second version, the algorithm can determine the parameter k automatically by applying the concept of gap statistics [TWH01]. This method requires the definition of a minimum and a maximum value for k but no further interaction with the user is required. The gap statistics estimates the best value for k by applying K-Means to different values of k and analyzing the quality of the clustering. Another method for automatically defining k could be the elbow method [Gov]. In this method, the user must plot various possible values of k and their performance with regards to the resulting clustering. Then, the user identifies the elbow of the resulting line that represents the best value for k. Due to the mandatory user interaction, we decided to omit the elbow technique. The performance of the K-Means algorithm depends heavily on the definition of k and the user may not have the expertise to determine the number of distinct situations a priori. Further, the K-Means algorithm always assigns all observations to an existing cluster and cannot identify noise, which could negatively affect the performance of the framework. Therefore, we additionally integrate two density-based clustering algorithms into the Situation Detection component to reduce these drawbacks.

We select DBSCAN and OPTICS as density-based clustering approaches. Neither method requires a number of clusters as input. Instead, DBSCAN requires the definition of `min_samples`, which specifies the minimum number of observation samples to form a cluster. Additionally, an ϵ (`eps`) value is required that defines the neighborhood of a data point in which at least `min_samples` must be found to classify that data point as core-point. For the definition of ϵ the user needs domain knowledge and it has a great impact on the identified cluster structure. OPTICS needs the parameter `min_samples` which is the number of data points in a neighborhood, to consider this point as core-point. Also required is the parameter `min_cluster_size`, which is the minimum number of data points required to form a cluster. The user can determine both values by considering how long a situation is usually active in the use case and how many observations are sent to the framework. Both density-based clustering algorithms can classify observations as noise, which could happen when the use case observes a new situation for a short time.

One important point that the user of the framework must keep in mind is data management. Since the use case continuously sends observation data, the amount of data is constantly increasing. So far, we have not implemented any feature to reduce the amount of considered data in the decision making, which may lead to errors due to memory limitations. To reduce the amount of considered and stored data, the framework needs to determine what information will be important in the future and what information can be omitted without negatively impacting the future performance of the framework. One option is to set a maximum number of data points considered, but this could result in sparse situations being forgotten. Techniques for reducing an ever-increasing amount of observation data can be found in the literature. For example, Kang et al. [KCP20] research on the required knowledge of robots about their environment to reduce the probability of collisions due to estimation errors regarding other robots. Such a technique could be easily added to the Situation Detection component as future work to prepare the framework for long-term executions as well.

7.5.4 Strategy Selection

The Strategy Selection is the second component of the framework, that is invoked by the Coordination component as depicted in Figure 7.3. This component is responsible for selecting the most promising adaptation planning strategy for the use case with respect to the current situation. This assumes, of course, that the use case supports different strategies and that the user configured them for selection. This functionality is based on the No-Free-Lunch Theorem for optimizations [WM97] and the idea of situation-dependent behavior of adaptation planning systems. Hence, the goal is to select the strategy that seems most promising for the current situation. To do this, the framework uses the experience gained from previous executions of the strategies in similar situations. However, which algorithm performs best in a new situation is not known a priori. Therefore, the framework must test the available strategies and start a new round of learning for that situation. A general definition of the algorithm selection problem can be found in [SM09]. In the following, we explain the general workflow of the Strategy Selection and refer to Algorithm 3.

Algorithm 3: Pseudocode workflow of the Strategy Selection component.

```
Input: DDM, current strategy, number of optimization attempts already
       performed, all observations for the current situation
1  strategy ← current strategy
2  if number of optimization attempts < DDM.min_optimization_attempts then
3      return strategy
4  else
5      exceed_counter ← 0
6      for observation within DDM.window_size do
7          if thresholds exceeded then
8              exceed_counter++
9      if exceed_counter >= DDM.threshold_exceeds then
10         if all strategies already executed for this situation then
11             strategy ← best performing strategy in history
12         else
13             strategy ← next strategy determined in DDM
14 return strategy
```

Similar to the Situation Detection, this component also receives the DDM as input. Additionally, the Coordination component sends the currently active adaptation planning strategy, the number of optimization attempts already performed for this strategy, and all available observations for the current situation. These observations contain the performance measures of the adaptation planning strategy and form the basis for the decision logic. First, the Strategy Selection sets the currently active strategy as the selected strategy since it assumes that no changes need to be made by default (line 1). Then, the component checks whether enough optimization attempts have been made to decide whether the strategy should be changed. We decided to provide a fixed initial period during which multiple optimizations of the parameters are performed before considering a strategy infeasible for this situation. If the actual number of optimization attempts has not reached the minimum number of optimization attempts defined in the DDM, it means that the Parameter Optimization component might need more time to optimize the parameters of this strategy, this component then returns the currently active strategy (lines 2-3). If the required number of optimization attempts has already been reached (line 4), this component can select another strategy if the current strategy does not meet the performance expectations (lines 5-8). To do this, the component analyzes the performance of the strategy in the last observations with respect to a defined threshold and counts the number of times the threshold is exceeded. The actual number of analyzed observations is determined using the *window_size* in the DDM. The component provides two ways to define these threshold: (i) hypervolume threshold and (ii) individual value thresholds. Full details of both methods are provided later in this section. After the component determines the number of threshold violation in the last observations, it checks whether this number is above the predefined maximum allowed threshold violations (line 9). If this holds, the component proceeds and selects a new strategy (line 10). It then checks to see if all strategies for that situation have already been executed and if so, it selects the strategy that resulted in the best performance measurements (line 11). Thus, the component computes the HV of performance measurements for each observation within the *window_size* and all strategies and selects the strategy that yields the highest average HV. In the event that at least one strategy defined in the DDM was not executed for this situation, the Strategy Selection retrieves one of these strategies from the DDM (line 13). This triggers a trial-and-error phase in this component, since the decision cannot be based on experience and the component is forced to try new combinations. Finally, the component returns the selected strategy to the Coordination component (line 14).

The Strategy Selection component provides two possibilities to determine whether an algorithm meets the performance expectations or should be modified. In both mechanisms, the component counts the number of threshold violations and compares them to the allowed threshold violations specified by the user. The first method the component offers so far is the HV threshold method which reduces the performance measures to a single score. In this case, the component computes the HV metric (c.f. Section 3.3) and compares its value to a user-defined threshold. To calculate the HV, the user must specify reference values for each performance measure in the DDM. These reference values can either be defined out of range for the performance measure in question, or set to a value within the range that should never be dropped below. If the reference value is defined within the value range and the actual value falls below this value, the HV is defined as zero regardless of the other performance measures. However, the downside of this method is that it weights measures with a larger value range more heavily, so the user should apply a normalization mechanism before sending the performance measures to the framework. Still, the advantage of this method is that the performance of the overall adaptation planning system is condensed into one metric and the user only needs to specify one threshold value.

The second possibility to determine whether to change the currently active adaptation planning strategy is to set individual value thresholds. This method requires the user to define individual thresholds for each performance measure of the DDM that should never be fallen below. Whenever one of the performance measures falls below this threshold, the Strategy Selection component counts this as a threshold violation, regardless of any possibly perfect performance of the other measures. This method allows the user to have more impact on the individual performance measures and value ranges of these measures are less important. The user can even rule out performance measures having an impact on the strategy selection by setting the threshold out of the value ranges. Similar to the other components of the framework, the user can develop a customized version of the methods used in this component. Additionally, the user can easily extent the functionality of this component due to its modular design. For instance, Machine Learning techniques such as Random Forests [GBBGP21] can be integrated to learn a model for the Strategy Selection. This learning could use historical observation data to computes features as basis for the learned model. As the framework detects new situations, the model should be retrained to also cover decisions for the new situation once sufficient observation data has been collected.

7.5.5 Parameter Optimization

The last component to be presented is the Parameter Optimization component depicted in Figure 7.3. The Coordination component invokes this component when a new strategy is determined, the situation changes, or the performance of the strategy decreases with respect to the performance measures of the use case. This component uses optimization techniques to determine the best performing parameter setting for the selected strategy. According to the No-Free-Lunch Theorem [WM97], the choice of the optimization algorithm depends heavily on the use case being optimized. At the moment, this component uses Bayesian Optimization to optimize the parameter setting as it performed best in our preliminary study on situation-aware optimization of platooning coordination strategies in Chapter 6. However, the desired technique can be easily replaced due to the modular nature of the framework. The decision to use the Bayesian

Optimization is based on our platooning coordination use case, as our study of situation-dependency showed that Bayesian Optimization is best s best for this use case [LNH^{+}21].

So far, the Parameter Optimization component applies Bayesian Optimization to determine a new set of parameters for the current strategy. For this computation, the component uses historical observation data of the same situation and strategy combination. The Coordination component is responsible for providing only relevant data to this component. If the situation-strategy combination has not changed since the last invocation of this component, the Bayesian Optimization integrates only the last observation into the optimization model to computer new parameters. If either the situation or the selected strategy has changed since the last invocation, the optimization model must be re-trained using historical data of the new situation-strategy combination, if available. This allows the Parameter Optimization to react to the current situation and strategy and learn from previous decisions.

The Parameter Optimization component returns the new parameter set for the strategy to the Coordination component which forwards the adaptations to the use case. The use case executes these adaptations and collects new observations, that is, performance data for the new parameter settings, and sends them to the framework. In the next round of execution by the framework, the optimization technique receives this performance data and performs the next optimization step to further optimize the strategy parameters.

When choosing which optimization technique to use, the user of the framework must keep in mind that the algorithm must be able to learn from previous decisions. It should also be noted that the algorithm must process new incoming observations on the fly and does not need to be fully trained for every new observation. Finally, the overhead to completely retrain the model when a new situation occurs or the selected strategy changes should be kept to a minimum. In the best case, the user should choose an optimization technique whose optimization model can be extracted and reloaded when the component needs to handle a new situation and strategy combination. This would limit the time required to completely re-train the model for each change in situation and strategy. For future extension of the framework, the general meta-heuristic search algorithm Stepwise Sampling Search (S3) [Noo15] could be tested for faster optimized parametrizations.

7.6 Use Case-specific Adapter of the Framework

All the components of the framework are designed to be generically applicable to a variety of use cases enabled by the DDM definition of use-case specific characteristics and an adapter that manages the connection between use case and framework as described in the following. As mentioned earlier, the framework is modular and consists of components that users can adapt to the use case depending on their requirements. Nevertheless, all components are designed to handle any kind of data from a use case as long as the data and optimization goals are defined in the DDM. This section briefly summarizes the required user actions to apply the framework for any use case.

Figure 7.6 provides an overview of the architecture of the adapter required to connect the framework to any use case. The self-aware optimization framework is shown at the top, while the use case consisting of the two lower levels (see Section 7.3) is shown at the bottom of the figure. The center of the figure presents two adapter components which are used to connect the framework and the use case: (i) Data Preprocessing and (ii) Adaptation Executor. The framework provides two interfaces that enable general applicability of the framework as they are implemented with Representational State Transfer (REST) Application Programming Interface (API)s. Further, the DDM defines the data sent through these APIs and provides all the necessary information to interpret the parameters, make adaptation decisions, and send adaptation actions. The API on the left receives the observation data, while the API on the right provides the possibility to retrieve adaptation decisions for the use case. We decided to further abstract the data handling from the use case and include an additional Data Preprocessing component. This component receives raw monitoring data from the use case, preprocesses this data, and potentially calculates additional aggregate metrics that may be required to assess the performance of the use case. Due to the REST API, the whole component can be replaced with a customized version to fit the desired use case. The Adaptation Executor component, depicted on the center right of the figure, retrieves the adaptation decisions from the framework and converts these into specific adaptation actions for the use case. This component also depends on the use case and the user must customize the component to the requirements of the new use case. Since both adapter components handle data transfer to and from the framework, the use case dependent implementation effort should be minimal. If the use case already provides the possibilities to send monitoring data and retrieve adaptation decisions, these adapter components may not be necessary. However, we have chosen to provide a template for such components as they represent another level of abstraction and, thus, reduce the computational effort in the use case.

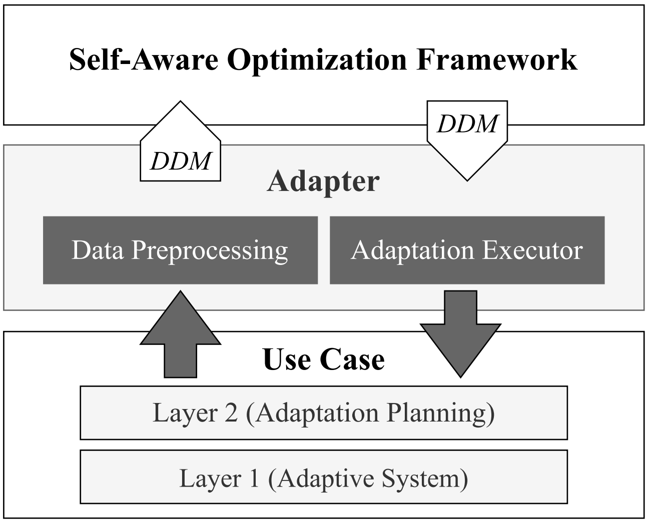

Self-Aware Optimization Framework
DDM
DDM
Adapter
Data Preprocessing
Adaptation Executor
Use Case
Layer 2 (Adaptation Planning)
Layer 1 (Adaptive System)

Figure 7.6: Use case adapter for the generic self-aware optimization framework. The use case with its two layers adaptive system and adaptation planning are depicted at the bottom. It communicates with the Framework by sending observations and retrieving adaptation actions. Additional Data Preprocessing and Adaptation Executor components can provide a further abstraction level.

7.7 Fairness-Ensuring Adaptation Planning Strategies

In the previous sections, we defined our self-aware framework for optimizing adaptation planning strategies. In the Foundations part of this thesis, we motivated this framework and our selected use cases by several properties of applied strategies such as the situation-dependence, fairness, and uncertainty in adaptation planning. Motivated by the No-Free-Lunch Theorem [WM97], our framework directly integrates situation-awareness as part of its workflow. The remaining properties are fairness and uncertainty from which we address fairness in this section. Similar to Rescher, we see fairness as “dividing goods or bads on the basis of general principles that pertain to everyone alike” [Res02, p.13]. We address properties in another way, that is, by introducing specialized fairness-enabling adaptation planning strategies as introduced in this section, or the definition of utility functions to address uncertainty. These proposed strategies form a contribution in SAS on addressing fairness aspects in such systems and can be used by the framework by selecting the most appropriate fairness mechanism according to the current situation.

In this section, we still focus on the ITS domain and address the fairness of positive effects when driving in a platoon. However, the contribution proposed in this section to address fairness within adaptation planning strategies can also be generalized to other domains. We propose six mechanisms to enable a fair intra-platoon positioning, which is an important aspect of platooning coordination as the diverse positions inside a platoon formation are directly related to the advantages that can be drawn from driving in a platoon. For instance, the lead vehicle experiences reduced fuel savings of around 5% while follower vehicles experience fuel savings of up to 15% [BSC+12]. We address these unfair circumstances of platoon participation by proposing approaches for indirect compensation of negative effects between platoon members. As already mentioned, these mechanisms refer to the proposed framework, as they can be applied in a use case to address fairness aspects and the most appropriate mechanism for the current situation can be selected using the framework. This section is based on our publication "A Comparison of Mechanisms for Compensating Negative Impacts of System Integration" in the Future Generation Computer Systems Journal [LKS+21].

The use case for the mechanisms pictures a road with at least two lanes and one-way traffic flow. On the road, only one platoon and non-platooning traffic are simulated. Even if there are more than two lanes, the mechanisms only occupy the right lane and the lane left to it at times. The latter will be referred to as the left lane even though there could be additional lanes left of it. On top of that, we state the following simplifying assumptions to limit the complexity:

- All non-platooning cars drive faster than the platoon and no overtaking of traffic cars is considered.
- All vehicles inside the platoon have the same vehicle type. Otherwise, differing acceleration or braking performances would lead to delays or make adaptations of the inter-vehicle spacing inevitable.
- We dismiss all kinds of limitations stemming from currently prevailing legal norms. Not only is the concept of platooning not yet legally enforceable in most jurisdictions given the low inter-vehicle spacing required for it, but also overtaking on the right lane as utilized by some strategies is forbidden in countries such as Germany.
- We assume that the vehicle to vehicle communication works flawlessly and explicitly exclude safety considerations as these are usually part of the communication and lower layers of platooning technology.

In the following, we introduce six mechanisms that rotate the position inside the platoon among all platoon members to equally split possible negative effects.

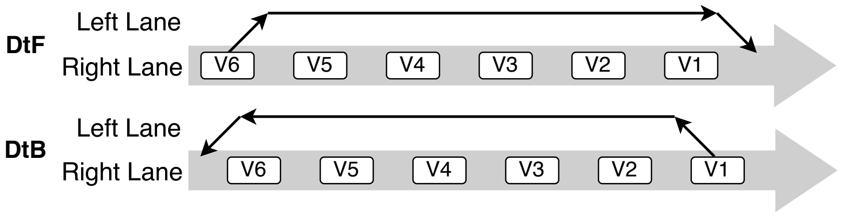


Figure 7.7: Illustration of the Drafting to Front (top) and to Back (below) mechanisms for addressing fairness within a platoon. The first or the last vehicle, respectively, leave the platoon to overtake or fall back and proceed as leader or at the back of the platoon.

The first two methods handle a single vehicle at a time while the last four methods establish a constant rotation of all vehicles which is inspired by the Belgian Tourniquet.

7.7.1 Drafting a Single Vehicle to the Front (DtF)

In this method, one of the following vehicles, for instance, the last vehicle (V6) of the platoon, overtakes the platoon and takes over the lead (see Figure 7.7). Therefore, it temporarily leaves the platoon and switches lanes once it is safe to do so. After the lateral movement is completed, the car overtakes and increases its desired speed. When the overtaking car is in front of the platoon, it decelerates again. Once the leader (V1) has been passed by a safe margin, the overtaking car (V6) switches back to the right lane. Finally, the overtaking car (V6) establishes itself as the new leading vehicle. With this maneuver, it is possible to either always rotate the last vehicle to the front or select a specific vehicle to become the new leader if there is a compensation model that tracks leading time over multiple platoons. The strengths of this method lay in a low disturbance of other traffic vehicles, as overtaking is performed rather quickly.

7.7.2 Drafting a Single Vehicle to the Back (DtB)

In this method, depicted at the bottom of Figure 7.7, the leading vehicle (V1) leaves the platoon, switch lanes and then let the platoon pass before queueing up behind it as the new tail of the platoon. First, the second car (V2) in the platoon is assigned the lead role. Then, the previously leading car (V1) switches to the left lane as soon as safely feasible. Once it has completed the lane switch, it will start to fall back by reducing the desired speed. After the overtaking car (V1) has fallen behind half of the platoon, it picks up pace again to rejoin the platoon with similar speed to the rest of the platooning vehicles. This prevents the creation of a large gap at the end of the platoon. When the drafted back vehicle has the desired distance behind the tail of the platoon (V6), it switches back to the right. Beneficially the singled out vehicle (V1) will not have to use more fuel by increasing its speed and overtaking; instead, it can coast until the platoon has passed. However, we expect that the utilization of a slowing car on the left lane will force traffic vehicles to brake more often.

7.7.3 Belgian Tourniquet (BT)

This technique originates from professional cycling, where usage of the slipstream effect is essential to save energy. To apply this technique for platoons, we separate into two groups across two lanes and establish a constant rotation. The method is similar to DtF; however, the next vehicle starting an overtake is always the last one of the original platoon, that is, V3 in Figure 7.8. It starts the overtake whenever there is enough space on the next lane, i.e., it does not wait until the currently overtaking vehicle (V6) is deployed as the new leader. The leader of the overtaking platoon (V6) is meant to overshoot the leader of the overtaken platoon (V1) and reduces its speed after it has rejoined the right lane. As soon as the switch to the right lane is finished, this car (V6) becomes the new leader as long as no other car switched to the right lane in front of it. This behavior is favorable for this approach because the following cars on the overtaking lane would be forced to brake and accelerate once again otherwise. Using this method, the cars on both lanes make use of platooning and because of the constant rotation, the benefits are approximately equally distributed at all times. Though, one possible drawback of this method is the starting sequence. As the initial situation includes a platoon driving on the right lane, the overtaking platoon needs to be formed from scratch. First, the last vehicle (V6) starts its overtaking maneuver before the next vehicles follow (V5) and (V4).

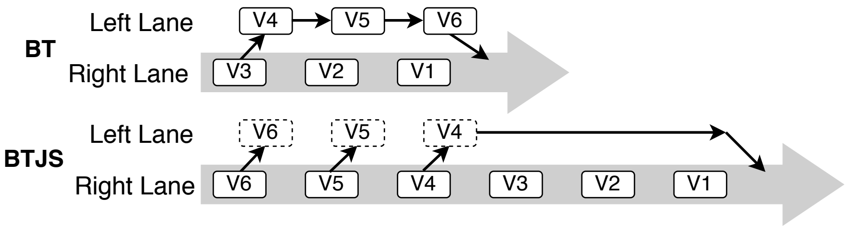


Figure 7.8: Illustration of the Belgian Tourniquet (top) and Belgian Tourniquet with Jump-start (below) mechanisms for addressing fairness within a platoon. These mechanisms are inspired by professional cycling where the so called Belgian Tourniquet is often applied.

7.7.4 Belgian Tourniquet Jump-start (BTJS)

Since it takes a while to get the continuous rotation of the BT properly running, we also consider an alteration of it with an improved starting procedure. In this instance, the trailing half, that is vehicles (V4) to (V6) in Figure 7.8 of the platoon, switches lane synchronously once enough space is available to jump-start the rotation. Afterward, the performed actions are the same as in the standard version of the BT as long as the flow is not interrupted. If traffic interferes such that no overtaking car remains, the rotation restarts with the same multi-car switch strategy. This alternative option should allow for an accelerated start to the platooning on the faster lane since the platoon is instantly split in half with the vehicles switching lanes already having a short gap to each other. Consequently, it should also lead to more overtaking maneuvers being performed in higher traffic densities as interruptions of the procedure will not be as costly given the rotation's faster restart. On the downside, it can not be guaranteed that the benefits are always distributed equally anymore because the timing of perturbing traffic determines when the strategy switches from the last vehicle of the platoon starting the overtake to a full restart with multiple vehicles. In the second scenario, the vehicle at the back of the original platoon, e.g., (V6), only takes the leading position after the vehicles in front of it, switching lane at the same time, e.g., (V4) and (V5), have done so, even though those have already led.

7.7.5 Reversed Belgian Tourniquet (RBT)

Akin to the distinction made between the methods DtF and DtF, we also propose a reversed version of the BT, depicted in Figure 7.9. Instead of the last vehicle of the platoon being the next to start an overtake, the leading vehicle (V1) is the next to switch lanes and fall back once enough space is provided. Now, the second car in the platoon (V2) is the leader as long as it drives on this lane. If this car starts its fall back procedure and switches to the left lane, the leader role is assigned to the next car in the platoon (V3). From there on, the rotation continues as the new leader of the original platoon (V3) will also follow suit once safely possible and so on. We hope this results in a more energy-efficient procedure since acceleration actions will only be performed while driving in the slipstream of a car in front. However, this also makes the implementation a bit more complicated. A vehicle falling back will have to pick up the pace again once it is getting close to the end of the platoon in order to adapt speed and assure the maintenance of a close gap after rejoining the platoon. For this process not to impact the smooth flow, it is critical to pick the speed and

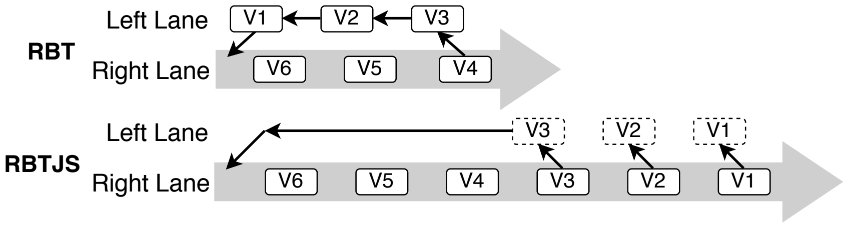


Figure 7.9: Illustration of the Reversed Belgian Tourniquet (top) and Belgian Tourniquet with Jump-start (below) mechanisms for addressing fairness within a platoon. These mechanisms are inspired by professional cycling where the so called Belgian Tourniquet is often applied.

the gap distance for the vehicles dropping back carefully. After all, it is a lot easier to maintain safe distances by braking of trailing vehicles rather than acceleration of preceding ones, as braking is more instant. However, suppose additional braking maneuvers have to be performed by a vehicle falling back. This vehicle cannot adapt its speed accordingly before reentering the platoon lane. Therefore, it will create a bigger gap at the end of the platoon that will also further complicate the rejoining of subsequent vehicles.

7.7.6 Reversed Belgian Tourniquet Jump-start (RBTJS)

Similar to BTJS, we also examine an alternative version to the RBT, jump-started by the preceding half of the platoon, that is vehicles (V1) to (V3) in 7.9, switching lane simultaneously. The switched vehicles will first take a short time to increase their inter-vehicle gaps before starting the usual procedure of the RBT. If the rotation is blocked by non-platooning traffic, and no platooning car is overtaking, the jump-start procedure re-starts.

Discussion of the Relation to the Framework

In the Foundations part of this thesis, we motivated the proposed framework by several properties of applied adaptation planning strategies such as the situation-dependence, fairness, and uncertainty aspects. Fairness is an important aspect in the platooning use case as the diverse positions inside a platoon formation are directly related to the advantages that can be drawn from driving in a platoon. For instance, the lead vehicle experiences reduced fuel savings of around 5% while follower vehicles experience fuel savings of up to 15% [BSC+12]. In this section, we proposed six mechanisms that address fairness in the platooning use case by switching the different positions inside a platoon and by this balancing negative and positive effects of platooning among all participants. We consider these strategies as adaptation planning strategies that—according to the No-Free-Lunch Theorem [WM97]—need to be selected carefully with considering the current situation. Hence, our framework can be used to automatically select the most promising fairness mechanism and enhance the fairness of the overall use case. We consider the direct integration of fairness mechanisms within the framework as not meaningful as fairness aspects of different use cases are too diverse. Hence, the contribution of this section is the definition of adaptation planning strategies ensuring fairness to show the possibility of addressing fairness in such mechanisms.

7.8 Addressing Uncertainty in Adaptation Planning Strategies

Similar to the previous fairness aspects in self-adaptive systems, we also motivated the ITS domain using uncertainty aspects in such systems in Chapter 6. Again, we apply our contribution of this section prototypical in the ITS domain but are convinced, that the approach can also be transferred to other domains. As already mentioned in the previous section, we now propose a methodology to address uncertainty in adaptation planning strategies. In line with the previous proposed fairness mechanisms, also the proposed contributions of this section relate to the framework as they can be selected and exchanged using the framework. Further, their parameters can be tuned using the parameter optimization component. We solely focus on an isolated view on uncertainty aspects in SAS and, hence, the integration of these aspects within the framework remain a main challenge for future work.

We still apply this methodology in the ITS domain and specifically study vehicle navigation problems in the presence of a dynamic environment and present CostSAVeR that plans routes for vehicles and incorporates the current traffic situation and refueling requirements. As CostSAVeR operates in a highly dynamic environment in which (i) the fuel prices are highly volatile and (ii) the traffic conditions vary spontaneously due to accidents, construction works or variability in the traffic volume, we are faced with high uncertainty in the planning of trips. To handle those circumstances, we design our system as a SAS [KRV^{+}15]. Hence, the system is able to modify its parameters and utility functions at run-time to adapt to the changes in its environment. In our case, we focus on an adjustment of the route for re-directing a vehicle (i) to a gas station and (ii) as a reaction to changing traffic conditions. This section is based on our paper "Utility-based Vehicle Routing Integrating User Preferences" which we published at the International Workshop on Pervasive Computing for Vehicular

Systems in Conjunction with IEEE PerCom 2021 [LHKK21b]. We now present CostSAVeR and propose utility functions that measure the cost-awareness of a route and model uncertainty that comes from changes in fuel prices especially for increased planning horizons, i.e., longer trips.

7.8.1 CostSAVeR

We designed our system as a self-adaptive system composed of a managing subsystem, called adaptation logic, which controls and adapts a managed subsystem [KRV+15]. The adaptation logic integrates a MAPE-K Control Loop [KC03] for controlling the adaptation. It incorporates the functionality for monitoring the environment, i.e., fuel prices, the current traffic flow, as well as possible traffic congestion, and the managed subsystems, for analyzing the situation, for planning the route, and for outputting the result to the user interface or an interface for autonomous vehicles. This procedure is performed in an iterative way to ensure an optimized route at any point in time. Adaptations in our use case depend on the route, on user-specified optimization constraints, or the dynamics of the traffic circulation. We integrate cost-efficiency as such constraint, hence, adaptations can be caused by changes in current fuel prices. However, for future work, it might be possible to add other factors for personalized routing.

We implemented CostSAVeR as a web-based prototype. As frontend client, an Android application[1] supports our adaptive navigation. It further supports real-time navigation based on the Google Maps navigation service. The backend receives the requests from the frontend, calculates the utility functions, and returns a ranked list of alternative routes to the frontend. In case of the web frontend, the calculation is performed once before the start of the journey. The Android app is able to adjust the route while driving. As the backend delivers a set of possible assessed routes so that the user can decide according to his preferences, e.g., using a specific route or focusing on gas station brands, our approach integrates the users in the loop.

In the following, we describe the self-adaptive route calculation system model based on the MAPE-K Control Loop, which is also depicted in Figure 7.10.

Monitor: In the monitoring phase, the required input is collected from the connected interfaces[2]: preference of the user (i.e., the goal of the user), origin and destination of the trip, average fuel consumption, price of the last filling,

[1]We published a running version of the Android app including a user guide on Zenodo: `https://doi.org/10.5281/zenodo.4067966`.

[2]We focus on the collection of data from the user interface and omit the integration of the On-Board Diagnostics (OBD)-II interface to collect data from the car.

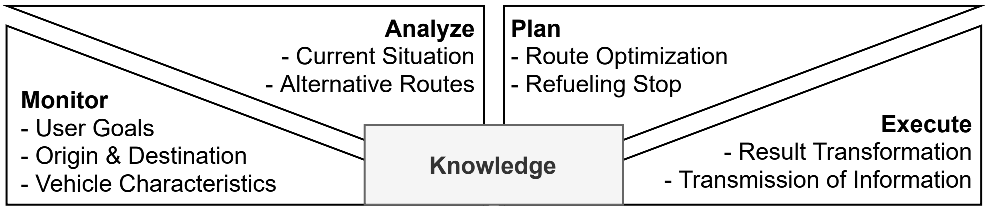


Figure 7.10: Model of our self-adaptive route calculation system CostSAVeR based on the MAPE-K Control Loop that integrates information from two sources: the standardized OBD-II interface for accessing the data interface of a vehicle and a user interface of a web/mobile application.

vehicle type, fuel type, and remaining driving range. Besides the user-related data, the system collects data about its environment, that is, alternative routes, traffic information, and fuel prices. Here, we use the Google Maps API for requesting routes, the Here WeGo API for retrieval of gas stations, and the Tankerkoenig API[3] for requesting current gas prices.

Analyze: Next, the system uses the collected data to identify the current situation and to analyze if the current situation requires an update of the route. Among others, triggering changes can be a new selection of user preferences, an unintentional change of the route by the driver during the trip, or a change in the current traffic situation. We analyze a form of proactive adaptation by planning refueling in advance, i.e., at the beginning of the trip, and omit run-time adaptation as a reaction to changes in the fuel prices. The following reasons motivate this decision. First, we want to avoid information overload, which occurs when constantly requesting up-to-date information of the routes or gas prices and the overhead of continually computing new routes due to the fuel prices' volatility. Second, as we assume uncertainty in the fuel prices of stations that are far away, an adaptation at run-time based on the current prices does not seem to make sense. This could cause the route calculation to constantly change the route and, in the worst case, introduces more detours than a decreased fuel price could compensate. Finally, a reliable evaluation of adaptive behavior at run-time and the integration of all the uncertainty factors, among others real traffic flow, volatile gas prices, and individual goals of the driver, inside a simulation are hardly feasible.

Plan: Afterwards, the system performs the calculation of the possible routes and the optimization by determining the quality of the routes. Therefore, the planning component uses the information retrieved in the monitoring step combined with the determined situation of the analyze step. In case the desti-

[3]https://www.tankerkoenig.de/

nation is reachable with the remaining fuel, no refueling is scheduled. Using the information about the remaining driving range, the algorithm checks if the destination is reachable with the remaining fuel considering an additional safety margin. If this is not the case, the route is updated with gas station locations and additional station information, such as brand and prices. We assume, that one refueling stop is enough to reach the destination. The combination of information about routes and gas stations forms new routes with potential detours for reaching a gas station. Each of these routes is assessed using utility functions and the planning component returns a ranked list of routes.

Execute: The main objective of the execution component is the transformation of the list with alternative routes such that the user is able to understand the information. Further, the transmission of information is the second responsibility. The transmitted information includes the following parameters: route, travel time in minutes, distance in kilometers, total costs, utility value, gas station information (brand, location, price, distance to the station).

Knowledge: Finally, the knowledge base contains all monitored information such as user goals, environment observations, and vehicle characteristics. Further, it stores the analyzed situations, planned routes, and utility functions used for assessing possible routes.

7.8.2 Utility Functions

Utility functions represent one way to evaluate which adaptation from a search space fits best to perform self-optimization [WTKD04]. Compared to advanced machine learning-based procedures or approaches that integrate mathematical or statistical optimization algorithms, the decision making process using utility functions is more lightweight and can be calculated faster. We introduce six different utility functions as a representative set of three categories: (i) integrating measures of the gas price, distance, and duration; (ii) coping with uncertainty of volatile gas prices; and (iii) selecting the nearest or completely random gas stations. Each utility function calculates a value per route.

Price-aware: The first utility function integrates the already paid costs for the remaining amount of fuel and the new refueling costs. Equation (7.2) presents the calculation of the estimated costs using the price per liter of the last refueling (p_{last}), the number of liters remaining ($l_{remaining}$), the current price per liter (p_{cur}) at the desired station, and the number of liters required (l_{new}) for the planned tour.

$$cost_{est} = p_{last} \cdot l_{remaining} + p_{cur} \cdot l_{new} \tag{7.2}$$

For using the estimated costs as utility, we define a vector presentation of the estimated costs for all routes $cost_{est}[i]$ with i being the index of the i-th route. We use a modified version of the min-max scaling formula [ZC18] to calculate a ranking. Thus, Equation (7.3) normalizes the values to the bounds $[0, 1]$ so that the highest costs have a utility of 0 and the smallest costs a utility of 1, with x being a variable for the estimated costs:

$$x[i]_{reversed} = \frac{x[i] - \max(x)}{\min(x) - \max(x)} \tag{7.3}$$

By integrating the described cost calculation and the min-max scaling, the price-aware utility function is defined as:

$$U_{Pr} = \frac{cost_{est}[i] - \max(cost_{est})}{\min(cost_{est}) - \max(cost_{est})} \tag{7.4}$$

Duration-/Distance-aware: This utility function (Dur/Dist) also uses min-max scaling and includes the duration and distance into one function by forming a weighted sum as defined in Equation 7.5. Again, the reversed min-max scaling of Equation (7.3) is used. The developer defines the weights w_i, which must sum up to one. Considered attributes i might be costs or duration.

$$U_{DD} = \sum_i w_i \cdot x[i]_{reversed} \tag{7.5}$$

Since the other utility functions mainly focus on the cost attribute, weighting every attribute equally in this work will show the other attributes' impact on the solutions. These weights can be changed in the future to get different solutions but are fixed for our work.

Volatility-aware: Since gas prices are highly volatile, the *volatility-aware* utility function tries to minimize the uncertainty of price changes. Based on the assumption that the probability of a change of the price at a gas station increases with a larger distance towards the gas station, the idea is to reward closer gas stations with a utility bonus. The price-aware utility function defined in Equation (7.6) is calculated and a bonus is added concerning the distance of the station ($d_{station}$) from the distance to the destination (d_{dest}). We introduce the weights are called α for weighting the utility of the price-aware utility function and β for weighting the added bonus.

$$U_V = U_{Pr} * \alpha + \left(1 - \frac{d_{station}}{d_{dest}}\right) * \beta \tag{7.6}$$

We set the parameter α to $\frac{3}{4}$ and the parameter β to $\frac{1}{4}$. The reason for this weighting is to still let the cost factor dominate the function. Otherwise, only gas stations that are very close to the origin might be chosen.

Penalty-aware: The *penalty-aware* utility function tries to minimize unforeseen changes in prices. In contrast to the volatility-aware utility function, it uses the required time it takes to reach the gas station ($t_{station}$) instead of rewarding closer distances. A punishment p is added to the price-aware function for each period ($period$) it takes to reach the station as defined in Equation (7.7).

$$U_{Pen} = U_{Pr} - \left\lfloor \frac{t_{station}}{period} \right\rfloor * p \tag{7.7}$$

We set the $period$ to a value of 1800 seconds and the value of p to 0.05.

Nearest Station: With this utility function, we model behavior to refuel at the closest gas station. Therefore, we use the start location for a distance-based search for the closest gas station and calculate the costs according to Equation (7.2). In the evaluation, this utility function serves as a comparison for the other utility functions as it does not consider any route or cost-awareness.

Random: This utility function selects a gas station among the identified gas stations within the search radius randomly. This process is performed 30 times to receive average costs for comparison.

Discussion of the Relation to the Framework

Similar to the motivation of fairness aspects, we also discussed uncertainty aspects in SAS in our foundations and addressed the ITS domain by integrating refueling into vehicle navigation tasks in this section. Since our use case covers a highly dynamic environment in which (i) the fuel prices are highly volatile and (ii) the traffic conditions vary spontaneously we are faced with high uncertainty in the planning of trips. To handle those circumstances, we propose a SAS for vehicle navigation that integrates refueling actions. We address the mentioned uncertainty by the definition of utility functions hat measure the efficiency and cost-awareness of a route and model uncertainty. The proposed approaches of this section are part of the SAS research and the framework can be applied to enhance the overall use case. We consider the propose utility functions as adaptation planning strategies from which the framework can select the most promising ones for the current situation. Further, the framework is able to perform parameter tuning on the input parameters of the utility functions which is a promising advancement for handling uncertainty. Similar to the fairness aspects, we consider the direct integration of uncertainty aspects within the framework unfeasible as these are too diverse in various use cases. Hence, the contribution of this section is the design of a SAS addressing uncertainty and the definition of adaptation planning strategies to show the possibility of addressing uncertainty in such mechanisms.

7.9 Summary

In this chapter, we have proposed our main contributions, which focus on **Goal A**: *Self-aware optimization of adaptation planning strategies with particular attention to the field of ITS and logistics*. We answered **RQ A.2** and proposed a component-based self-aware optimization framework for adaptation planning strategies. With this framework, we reduce the manual effort required to optimize these strategies and developed a system that is generically applicable. We have also addressed **RQ A.3** by incorporating a situation-detection component directly into the framework. In contrast, we argue that fairness and uncertainty issues should be directly addressed in adaptation planning strategies. Therefore, we propose a set of adaptation planning mechanisms ensuring fairness for the platooning coordination use case to answer **RQ A.4**. We then design a set of utility functions, which can also be viewed as adaptation planning strategies, used by the framework to address uncertainty and answer **RQ A.5**.

Chapter 8

Optimization of Vertical Systems-of-Systems

In the previous chapter, we introduced a generally applicable self-aware optimization framework for self-adaptive systems. For this framework, we defined some restrictions such as the focus on an optimization of single systems. However, many application use cases can be classified as part of the systems-of-systems research area. Maier proposed a definition for the term systems-of-systems in his work from 1998 [Mai98, p. 271]:

> "A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possesses two additional properties: Operational Independence of the Components [...], [and] Managerial Independence of the Components[...]."

Operational independence of the components defines that the components when disassembled remain useful and can operate on their own even without the other components of the systems-of-systems (c.f. [Mai98, p. 271]). Managerial Independence of the components means that the components not only can operate on their own but do operate individually and remain operational no matter what the other components do (c.f. [Mai98, p. 271]). Hence, a system-of-systems consists of multiple complex components that collaborate to achieve a common goal and in its entirety forms an overall system. One example for a system-of-systems of the logistics domain is the VRP. As part of the VRP several TSP instances need to be solved to find a solution of the overall VRP. However, both parts of the overall problem remain functional even without the other components, since the VRP can also be solved with a random tour sequence, and the TSP does not have a direct backwards loop to the VRP. Hence, we consider this problem as a kind of vertical system-of-systems where the overall system triggers subsystems that feed back their solution to optimize the overall problem statement even further. We address the VRP in this section as prototypical use case for a complex system-of-systems in the logistics domain. Still, the contribution of this chapter can be generalized to other domains such as smart grid or intelligent computer networks.

In this chapter, we apply optimization techniques to tackle the VRP and study the effects of the system-of-system. With regards to the presented layered architecture in Section 7.3, we consider the approaches proposed in this chapter to be part of the second layer (adaptation planning layer) as they aim at optimizing the use case and, hence, plan adaptations for it. The proposed approaches can be applied once for each problem statement to find a valid solution, but are also able to be applied as part of a feedback loop that adapts the use case system in a possibly changing environment. With the knowledge about these kind of vertical systems-of-systems, the proposed framework from Chapter 7 can be extended to also cope with such complex systems and to be applicable to a broader range of use cases. However, we aim at handling the effects of optimization of systems-of-systems but the actual adaptation of our proposed framework is out of scope of this thesis. Hence, in this chapter, we address a particular type of the VRP called rVRP that integrates a large and diverse amount of real-world restrictions such as, besides others, capacities, pickup and delivery, and time windows. In cooperation with a logistics company, we define the specific problem statement to be as realistic as possible. Our scientific contributions in this chapter are two-fold:

- We define a two-stage strategy for tackling the systems-of-systems structure of the formulated rVRP including a (i) VRP-stage that assigns orders to vehicles and a (ii) TSP-stage that optimizes the tour for each vehicle.

- We propose a timeline algorithm within the workflow that modifies the planned tours in order to handle time windows and fixed pause stops to tackle additional constraints of the rVRP.

Further, we do not aim at planning the rVRP once at the beginning of the day, contrary we aim to be adjustable at any time to represent adaptive behavior using feedback loops and, hence, require algorithms having a fast time-to-result. This chapter is based on our paper [LKK^{+}21a] which is accepted for the Springer Applied Intelligence Journal and our technical report [LKK^{+}21b]. In the following, we first define the actual problem statement we derived in cooperation with a logistics company in Section 8.1. Afterwards, in Section 8.2 we propose our approach for the vertical systems-of-systems consisting of a VRP and a TSP stage. Section 8.3 presents our GA version and Section 8.4 proposes our customized ACO for both stages.

8.1 Problem Statement

In close exchange with our cooperation company, we defined the following real-world requirements for the rVRP we address in this chapter. Figure 8.1 illustrates the considered version of a rVRP as domain model. We define a *tour* t_j as the assignment of customer orders $o \in O$ to vehicles $v \in V$ and drivers $d \in D$: $t_j = (v_i, D_j, O_j)$ with a set of drivers $D_j \subseteq D$ and a set of orders $O_j \subseteq O$ assigned to tour t_j driven with vehicle v_i. The central goal is to find a set of tours $T = \{t_j\}$ so that all orders are assigned while minimizing the cost function defined in Section 8.2.2. A tour has a tour start and a tour end, each specified by a time and a location. The tour start time window defines the range in which the tour must start.

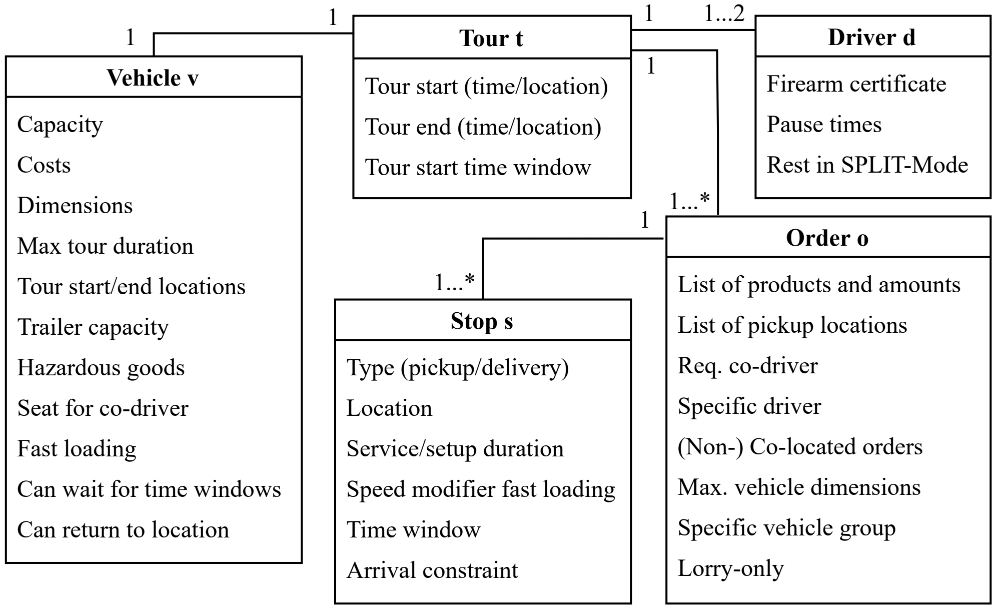


Figure 8.1: Domain model of the rVRP addressed in this chapter. The addressed rVRP consists of a set of tours as solution where orders are assigned to vehicles. Each order consists of at least one pickup and one delivery stop. Further, one or two drivers are assigned to a tour.

A *vehicle* v_i is always assigned to one tour t_j. Each vehicle has a capacity, defined by the number of pieces, volume (m^3), or weight (kg). The costs for using a vehicle are defined per hour, per kilometer, per tour, or per stop on the tour. The height, width, length (m), and weight (kg) of a vehicle define its dimensions. Each vehicle has a maximum tour duration, after which it must be at the tour end location. The tour start/end locations represent a list of locations at which the tour can start and a list at which locations the tour can end. The option for a trailer specifies whether the vehicle can carry a trailer. Since special properties of a vehicle are required for carrying hazardous goods, we integrate the possibility to specify whether a vehicle is able to carry those goods or not. This enables a valid assignment of orders to vehicles even for hazardous goods. Every vehicle may have a single seat for the driver or it may offer an additional seat for a co-driver. The fast-loading property states whether the vehicle can load and unload faster compared to other vehicles and contains a speed modifier representing the saved time. This property influences the required service and setup time and, hence, the time required at a specific stop. It provides the possibility to switch to a vehicle with fast-loading property. A vehicle might provide the possibility to wait for a specific time window at a given stop and might be allowed to return to a stop multiple times.

Besides, up to two *drivers* D_j for each tour t_j are determined. For some orders, a co-driver is required for loading and unloading bulky goods. Drivers might require a special training or certificate, for example a firearm certificate required for cash transports. By specifying the required certificates for drivers with regards to a specific order, a valid assignment of drivers to tours is possible within the optimization, making an additional post-processing step unnecessary. The legislation might prescribe a fixed set of pause times for each driver. Sometimes it may be possible to schedule pause times within a service, such as during a pickup or a delivery (called SPLIT-Mode). Otherwise, the pause time needs to be scheduled while driving between two stops.

Additionally, customer *orders* O_j need to be serviced during a tour t_j. Each order contains a list of products with the amount specified by quantity in pieces, weight (kg), or volume (m^3). For each order, one can specify one or more pickup locations, a co-driver requirement, or the assignment of a specific driver are possible. Orders that should or should not be delivered in the same tour can be defined as a list of (non-) co-located orders. Maximum vehicle dimensions, a vehicle from a specific vehicle group, or a lorry-only service can be required.

For each order, at least one *stop* s needs to be scheduled. Each stop is either a pickup or a delivery stop and has a specific location. The setup/service duration contains the time the vehicle stands still while loading or unloading. The speed modifier for fast loading vehicles defines the saved time during the setup/service duration if this stop is assigned to a vehicle with this property. Each stop has a time window that specifies at which interval the driver needs to arrive or finish the service.

8.2 Approach

This section describes our approach for tackling the rVRP. We introduce our two-stage strategy to address the systems-of-systems nature of the problem statement and present the cost function. Further, we propose a timeline approach to match given time windows and pause stops for each planned tour.

8.2.1 Two-staged Strategy

Due to the high complexity of the problem, and to address the systems-of-systems nature of the problem statement, we divide our approach into two stages as inspired by [CT10] and depicted in Figure 8.2. Hence, we propose to use one stage for the overall VRP and a second stage for the contained TSP. The VRP-stage cannot be solved without solving the TSP-stage which is the special characteristic of this vertical system-of-systems structure. First, we address the problem of distributing all orders, including pickup and delivery options, to the available vehicles which we refer to as VRP-stage. In this step, several assignment-related constraints such as order restrictions are addressed. However, many of the above mentioned constraints are sequence-dependent and, hence, the nested TSP instance for each vehicle needs to be solved. In the TSP-stage, the TSP-solver starts and solves an individual TSP instance for each vehicle. This stage computes the optimal sequence of stops of one tour with regards to the tour specific constraints. Therefore, we retrieve the actual stop-to-stop route from a route planning service and optimize the cost function by changing the sequence of stops. The solved TSP instances are then sent back to the VRP-stage that performs our Timeline algorithm presented in Section 8.2.3. The Timeline algorithm matches the predefined time windows and schedules required pause stops for each tour. Finally, the distribution can be rated with regards to the cost function explained in the next section and the algorithm decides whether the current distribution should be kept or discarded. Depending on the size of the VRP and the nested TSP instances, either exact (for smaller problem spaces) or heuristic approaches (for large problem spaces) can be used to solve the stages. Since we do not want to restrict applicability of our proposed system to only work for small TSP instances, we propose our heuristic approaches based on GA and ACO for both stages. We select these algorithms as they are nature-inspired heuristic algorithms which are commonly used to tackle the class of TSP and VRPs. Still, also other heuristic optimization techniques could be applied in the future to assess their performance.

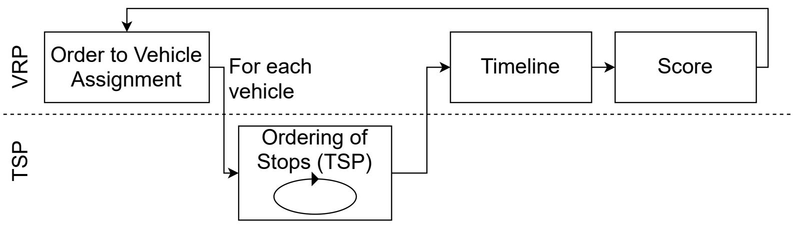


Figure 8.2: Overview of the two-staged strategy consisting of a VRP and a TSP-stage addressing the vertical system-of-systems structure. The VRP-stage assigns all orders to available vehicles and triggers a TSP-stage for each vehicle. Afterwards, the Timeline algorithm matches time windows and pause times before the VRP-stage rates the solution by calculating the score.

8.2.2 Cost Function

Since the rVRP addressed in this chapter exhibits a high diversity of constraints and restrictions, we propose to use a priority cost function for the evaluation of the generated solutions. We define this cost function using six priority scores as a classification of constraints and objectives discussed with our partnered logistics company resulted in six groups. Priority score means in this context that the first score value has a higher priority for the optimizer and will be minimized first, before the optimizer even recognizes the other following scores. The six scores are divided into three hard scores $[H_1, H_2, H_3]$ and three soft scores $[S_1, S_2, S_3]$. The hard scores assess the solution's feasibility and are handled as hard constraints, while the soft scores represent the solution's quality. The scores form a minimization goal for the optimization process.

In case the planned tour exceeds the capacity of vehicles and trailers, the first hard score H_1 sums up the exceeded capacity by subtracting the actual vehicle capacity (v_{cap}) from the planned vehicle capacity (v_{pcap}). Further, it adds a value of 100 score points for each fault in existing order restrictions such as a co-driver requirement (f_{or}) and a violation in order dependencies like co-located orders (f_{od}) where the operator # indicates the number of violations. We decided to use the multiplier 100 for the order restriction violation to be able to integrate the number of violations and capacity exceeds into one score. This is required as the order restrictions are counted as number of violations and the capacity exceeds are counted as difference of weights, volume, or the like, which naturally results in higher values and reduces the mathematical impact of comparatively low amounts of violations.

$$H_1 = \sum_{v \in V} max((v_{pcap} - v_{cap}), 0) + 100 \cdot \#f_{or} + 100 \cdot \#f_{od} \tag{8.1}$$

The second hard score H_2 deals with the pickup and delivery of orders (P&D), the entry order, and the vehicle assignment. First, the score checks whether the pickup is done prior to the corresponding delivery with 100 score points for each fault (f_{pd}). Afterward, the vehicle-specific tour start and end locations are examined and one score point is added for each fault (f_{se}). Here, a fault means that the vehicle is planned to start or end at another location than specified in the domain model. Finally, this score evaluates whether all stops that require a specific vehicle are serviced by such a vehicle (f_{sv}) and whether all planned returns to stops, that is, multiple stops at the same location, are allowed (f_{sr}). Any fault adds one score point to H_2.

$$H_2 = \sum_{v \in V} 100 \cdot \#f_{pd} + \#f_{se} + \#f_{sv} + \#f_{sr} \tag{8.2}$$

The hard score H_3 sums the seconds the tour duration (t_{dur}) exceeds the maximum duration (t_{maxdur}) and the planned tour end (t_{pend}) exceeds the end constraint (t_{end}). Since this score summarizes deviations from the plan in seconds and, hence, a plan miss by one minute already increases the score by 60 score points, we omit to add a further multiplier.

$$H_3 = \sum_{v \in V} max((t_{dur} - t_{maxdur}), 0) + max((t_{pend} - t_{end}), 0) \tag{8.3}$$

The soft scores evaluate the quality of the determined solutions. The first soft score S_1 assesses how good the solution matches each time window (tw) in the set of predefined time windows (TW). Since pause times can also be reduced to time windows that need to be met, we decided to handle pause times similar to time windows. Hence, when talking about time windows with regards to the score S_1, we always refer to the time windows that need to be met at the customer locations as well as the predefined pause times that need to be scheduled for the driver. This score sums up how many seconds the planned time window (tw_p) exceeds the given time window (tw_g). Therefore, the seconds the planned time window starts ($tw_{p,s}$) ahead of the given time window ($tw_{g,s}$) are calculated and added to the seconds the planned time window ends ($tw_{p,e}$) after the given time window ends ($tw_{g,e}$).

$$S_1 = \sum_{v \in V} \sum_{tw \in TW} max((tw_{g,s} - tw_{p,s}), 0) + max((tw_{p,e} - tw_{g,e}), 0) \tag{8.4}$$

The second soft score S_2 summarizes driven kilometers ($dist$), waiting times ($time_{wait}$), driving times ($time_{drive}$), and service times ($time_{service}$). Since these objectives form the main goal of the defined VRP in this work, we decided to integrate them into one score and, hence, assign the same priority to these objectives. Again we avoid to add a multiplier to these values since they are measured in km and seconds that tend to grow very quickly.

$$S_2 = \sum_{v \in V} dist + time_{wait} + time_{drive} + time_{service} \tag{8.5}$$

The last soft score S_3 refers to the delay of a driver starting its tour (t_{pstart}) after the defined start (t_{start}), the number of visited locations (loc), and the chain length (cl), that represents the number of stops to be serviced during the tour. This score integrates further soft constraints that are less important than the main objective goals in S_2 and is only assessed if several solutions perform equally well on S_2. Hence, this score is used to decide which solution performs best, if multiple solutions perform equally well in our main objective score S_2 and, hence, serves as tiebreaker.

$$S_3 = \sum_{v \in V} max((t_{start} - t_{pstart}), 0) + \#loc + cl \tag{8.6}$$

An example score value for a VRP solution that meets all capacity constraints, breaks one order restriction and sticks to all entry order and location-specific constraints can look like: Hard $[100, 0, 0]$, Soft $[120, 2919200, 1235]$. The H_1-value of 100 represents the order restriction fault of this solution, while H_2 and H_3 have a value of 0 indicating, that these constraints are all met. The S_1-value of 120 means that the vehicles of this solution break time windows by 120 seconds. The S_2-value 2919200 is the sum of all service, driving, and waiting times, while the last score (S_3) refers to the delay of starting times and the number of visited tours.

In contrast to the given example and in order to save computation time, we only calculate scores with lower priority (H_3, S_1, S_2, and S_3) if the previous hard scores are down to zero. Otherwise, the solution is considered to be infeasible if the hard constraints are not fulfilled, i.e., the hard scores are not reduced to zero. Since we implemented our score system as priority scores that need to be minimized, the first smaller value of a score level—starting at H_1 and ending at S_3—decides which of the two solutions performs better.

8.2.3 Timeline Algorithm

In this section, we introduce the *Timeline algorithm* to match the pause times and fit as many time windows of stops as possible (see Algorithm 4). Please keep in mind that we handle both, time windows at customers and pause times for drivers, equally in this algorithm and, for the sake of simplicity, call both of them time windows. This algorithm fits all pause times first and then tries to fulfill all time window requirements. All stops are shifted to fulfill all pause times and no pause time violations will be present after its execution.

Algorithm 4: Pseudo-code for the Timeline Algorithm.

```
   Input: Sequence of stops $seq$
 1 Initialize timeline $t$ with required pause times and customer time windows
 2 Buffer = tour start interval
 3 Penalty = 0
 4 foreach $s$ in $seq$ do
 5     Add $s$ as early as possible on $t$
 6     if TW not yet reached then
 7         Shift to right until TW is met or buffer is empty
 8         Reduce buffer by shifted seconds
 9     if (buffer is empty AND TW not yet reached) then
10         Add waiting time to $t$ until TW starts
11     else if TW already passed then
12         Increase penalty by missed seconds
13 return Penalty
```

This algorithm iterates over each sequence of stops, i.e., once per vehicle and tour, and calculates a penalty for the score. It first initializes the timeline with given start and end times of a tour, pause times, and time windows. Then, it iterates over the sequence of stops and places all stops as early as possible taking into account the sequence retrieved from the TSP-stage, its tour start interval, and its time windows. If the current timestamp is too early for the time window's starting time, the algorithm shifts the whole chain of stops (excluding the pauses) to a later starting point while keeping all previous time windows and the tour start interval. This also includes a recalculation of all previously placed stops by a defined amount of time regarding the start time of each stop. In case a shift is not possible, the algorithm adds a waiting time for the vehicle at this stop to meet the time windows. In case a pause time is reached while driving from one stop to another, the algorithm adds a pause on the route. If the SPLIT mode is activated and the algorithm schedules a pause time next to a service, the pause time is added to the service time. If it is deactivated, the full service is shifted after the pause. After the placement of all stops with time windows and pause times on the timeline, the scores are recalculated and feed back to the VRP-stage to judge the quality of the solution.

8.3 Genetic Algorithm

As part of our contribution to the systems-of-systems research regarding rVRP, we apply a customized GA on both stages of our approach. In the following, we describe the customized GA, that was originally proposed by [Hol92]. Figure 8.3 presents an object oriented illustration of the genome we used for our GA. Each genome contains a set of vehicles each representing a TSP instance. Each vehicle holds a list of orders. Each list is passed to the TSP-stage that determines the most beneficial sequence of this list.

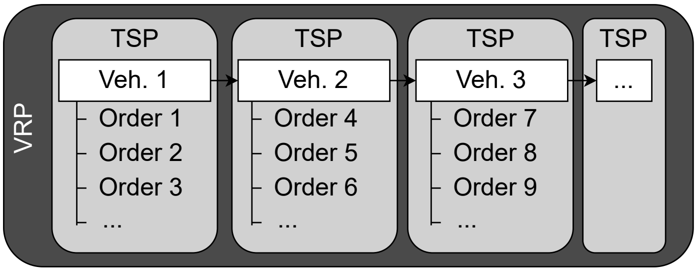


Figure 8.3: Illustration of an object-oriented genome representation of the GA. The VRP consists of multiple vehicles, each representing an individual TSP instance. The TSP instance further contains a sequence of orders assigned to a vehicle.

8.3.1 VRP-stage

Each GA starts with an initialization phase of the population from which the evolutionary process starts. These primal individuals form the population that is progressed in iterations, also called generations, by the execution of the algorithm by applying crossover and mutation operators. Hence, after an initialization phase for the population, each iteration of the GA performs four steps: (i) breed new individuals, (ii) solve the TSP-stage for each vehicle, (iii) calculate the score, and (iv) maintain population size. These steps are repeated until either a predefined number of unimproved iterations or a given computation time is reached as summarized in Algorithm 5.

Instead of initializing the population purely at random, we decided to create an initial individual by assigning orders to vehicles with regards to vehicle restrictions or co-location requirements. This allows the algorithm to form a meaningful initial population and provides a good starting point to kick-off the optimization process. Afterwards, the algorithm matches remaining orders to the vehicles based on stop-to-stop distances, that is, a stop that has the minimum distance to already assigned stops of a vehicle is assigned to this

Algorithm 5: VRP-stage pseudo-code for the GA.

```
Input: Orders $o$, vehicles $v$, max. iterations $i_{max}$, population size, mutation
       probability, crossover probability
1  Create initial individual and mutate it to initialize population
2  while unimproved iterations $(ui) < i_{max}$ AND current runtime < max.
    runtime do
3      while population size $< 2 \cdot$ initial population size do
4          Select two individuals
5          Apply a random crossover operator
6          Apply a random mutation operator with probability $p$
7          foreach $v$ do
8              Solve TSP-stage for offspring
9          Apply Timeline algorithm to match time windows
10         Add offspring to the population
11     Sort population by score and remove worst half
12     if improved best score then
13         $ui = 0$
14     else
15         $ui++$
16 return population
```

vehicle. Then, the algorithm improves this solution iterating over all stops and moving them to other vehicles in order to improve the average stop-to-stop distances for all vehicles. Then, for each required individual as specified in the population size, the GA selects one mutation operator from the list of available operators randomly and applies it to the individual to create the whole initial population. For each individual in this population and for each vehicle, the TSP-stage solves the stop sequence. Then, the Timeline algorithm is applied and the score per individual is used as its fitness value.

After the initialization phase—the creation of the initial population (line 1)—the VRP-stage GA iterates until one of the above mentioned termination criteria is met (line 2). Each iteration, that is, each generation, breeds new individuals until the population size has doubled (line 3). Therefore, the algorithm randomly selects uniformly distributed from three possible selection operators (that are introduced later) to breed a new individual from two parent individuals (line 4): (i) select two individuals randomly based on a uniform distribution; (ii) select two individuals randomly based on a predefined probability, where the individual with the best score has the highest probability; and (iii) a tournament selection where ten solutions compete pair-wise and the winner is selected for recombination. Then, the algorithm randomly selects a crossover operator from the set of provided operators and applies it on this pair of individuals (line 5). Afterwards, the algorithm mutates the new individual with a typical value in the literature for the mutation probability of $p_{vrp} = 0.5$ using a randomly selected mutation operator (line 6). With defining a set of selection and mutation operations and their random selection, we cope with the variety of constraints and aim at a higher diversity in the population. For each created individual, the algorithm forwards the TSP instances to the TSP-stage (described in Section 8.3.2) that solves this instance and returns ordered lists of stops (lines 7 and 8). Then, the algorithm applies the Timeline algorithm to match the given time windows (line 9). Finally, the algorithm calculates the score of the new individual and adds it to the current population (line 10). Since, the population size doubled during this iteration, half of the population needs to be discarded to match the predefined population size (line 11). Therefore, the algorithm sorts the population according to the achieved score and removes the worst half of individuals. This affects the next generation as only the best performing individuals are kept for recombination in the next iteration and therefore accelerates the convergence of the GA.

The crossover operators use two individuals for breeding a new offspring. Therefore, chains or parts of chains are copied from the parent individuals to the new individual. The remaining stops, that is, the sub chains that are not copied to the new individual, are assigned based on the stop-to-stop distance of each vehicle regarding already assigned stops. This means, for each stop that needs to be assigned evaluate the fit accuracy of the new stop by summarizing all stop to stop distances from the stop to be assigned to all stops already assigned to a vehicle. Then we select the vehicle with the lowest summarized stop-to-stop distances as we hope to achieve the least increase in the second soft score S_2 that incorporates driving distance and driving time, by this criteria. Since our problem definition includes diverse constraints, we define the following three crossover operators to breed new individuals. We explicitly avoid the selection of one best performing crossover operator as, in this way, we are able to maintain a higher diversity of the population and, hence, enhance the convergence speed of the algorithm:

1. The *OverlapCrossover* operator copies stops located on both parents to the new individual. Remaining stops are added to the vehicle of the new individual with lowest distance to existing stops of this vehicle.

2. The *ScoreBasedCrossover* operator copies the chain of the parent with lower costs to the new individual. The remaining stops from the other parent are added similar to the first crossover.

3. The *SelectionCrossover* operator selects one of the parents randomly and assigns the chain to the new individual. The remaining stops from the other parent are added similar to the other crossovers.

Mutation operators are used for breeding new individuals from a single parent individual and increasing the diversity of the population. For each individual that should be mutated, we select one mutation operator randomly. Since it is not guaranteed that a mutation operator produces a valid individual, we restart the mutation with another randomly selected operator in case the individual is invalid. Since our problem definition includes diverse constraints, we define the following mutation operators, each modifying the genome in a different way, aiming at a specific constraint. By providing this diverse set of mutation operators, we deal with the variety of constraints and are able to keep the diversity of the population as high as possible rather than focusing on a single mutation operator.

1. The *ClearVehicleMutator* removes all stops of a random vehicle and assigns them to other vehicles, based on a location and distance-based rating.

2. The *SwapVehicleMutator* swaps chains of two different vehicles, excluding the vehicle's start and end locations. Since we apply this operator at the VRP-stage consisting of multiple vehicles and their assigned orders, we consider it a mutation.

3. The *OutlierMutator* iterates through every vehicle's stop chain, selecting the stop pair that contributes most to the distance-based rating and moving it to another vehicle.

4. The *MoveOrderMutator* takes up to three orders of one vehicle and moves them to another vehicle, based on the distance rating. This behavior is repeated for a random number of times with a maximum of four times.

5. The *CloseToOtherVehicleChainMutator* selects a stop from a chain close to another chain, and moves the order for this stop to the nearby chain.

6. The *SavingsMutator* iterates over all stops of every vehicle and computes the highest saving in driving distance when moving one order to another vehicle. Additionally, predecessors and successors are moved to another vehicle if this reduces the distance. This avoids overlapping tours.

8.3.2 TSP-stage

The TSP-stage of the algorithm calculates the sequence and selects options, i.e., the list of possible locations, for each vehicle independently. Hence, the following description always captures performed steps for the tour of a single vehicle. At the beginning, the algorithm creates the initial population similar to the initialization of the VRP-stage by calculating a first valid individual. For this individual, the algorithm starts with a random stop and assigns the remaining stops based on the stop-to-stop distances, that is, the algorithm selects always the nearest stop compared to the last assigned stop. Then, the algorithm mutates this individual by applying randomly selected mutation operators to create the required amount of individuals for the initial population.

After the initialization phase, the TSP-stage GA performs similar steps compared to the VRP-stage GA. It iterates until the maximum amount of unimproved iterations are executed and breeds new individuals until the population size has doubled in each iteration. For the new individuals, the algorithm selects and recombines two randomly chosen parent individuals using a random crossover operator. Afterwards, the algorithm mutates the individual with a typical value in the literature for the mutation probability of $p_{tsp} = 0.5$ and a randomly selected operator and adds it to the population. As the population size doubled, the algorithm omits the worst half to accelerate convergence.

Again, the crossover operators combine two parents into one new individual. We define the following three crossover operators to breed new individuals in different ways and keep the diversity of the population high. The crossover operators in the TSP-stage are inspired by [HMS+17]:

1. The *RandomCrossover* randomly chooses the next possible stop from the beginning of the parents' chain while removing stops already contained in the population.

2. The *OrderedCrossover* performs a classical two-point crossover and combines the genome of both parents.

3. The *PartiallyMappedCrossover* works similar to the OrderedCrossover but assigns the remaining stops outside the interval at the beginning of the chain based on the indices of their parents which is the main difference to the one in the literature.

Additionally, we define the following mutation operators concerning the TSP-stage, inspired by related work [HMS+17]. Again, we decided to provide a diverse set of mutators and select random ones in each iteration to increase the diversity of the population.

1. The *ReverseMutator* reverses the sequence of all successive pickup and delivery pairs.

2. The *SimpleMoveMutator* moves one stop to another feasible position in the chain, taking into account the constraint of pickup-delivery order. The TourBegin and TourEnd nodes are protected and omitted from this mutation.
3. The *SimpleSwapMutator* swaps the positions of two stops on the chain.
4. The *MultiOptMutator* combines the previous two mutators and applies the SimpleMoveMutator or the SimpleSwapMutator up to three times.
5. The *NeighbourhoodSwapMutator* is similar to the SimpleSwapMutator, but it works based on distance improvement when swapping stops. It tries all possible swaps in the chain for one random stop and performs the swap with the highest distance improvement.
6. The *SavingsTSPMutator* selects the stop that produces the highest saved distance when moving it in the chain. The mutator calculates the delta of the distance concerning the whole chain and executes the move with the highest distance savings.
7. The *OptionsMutator* selects a random stop with at least one option and randomly replaces it with one of the other possible options.
8. The *OptionsChainMutator* rotates the options for the whole chain and replaces all stops with a possible option of this stop.

In summary, this section introduced our domain-adapted GA. First, we presented our object oriented genome presentation used and proposed two stages of this algorithm. Then, for each stage of the algorithm, we provide a domain-specific set of crossover and mutation operators that are randomly chosen in each offspring computation. These operators enable the algorithm to cope with the various constraints included in this work and aim at maintaining a high diversity of the population and a fast convergence speed.

8.4 Ant Colony Optimization

This section explains the two-staged ACO algorithm inspired by [DMC96]. We modified the classical ACO algorithm for both stages to accommodate for the complexity of the rVRP:

- We replaced the pheromone initialization by a heuristic concerning the actual stop-to-stop distances to kick-off the optimization from the first step onward.

- We use a deterministic ACO in the VRP-stage, this means that we start with an assignment of stops to vehicles based on the pheromone matrices. This helps to decrease the possibility of bad performing solutions at the start of the algorithm.

- The stops for pickups and deliveries are assigned in pairs, so that one vehicle needs to serve both stops in one tour. This prevents creating invalid solutions if pickup and deliveries would be assigned to different vehicles.

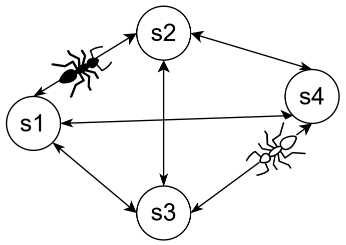
8.4.1 VRP-stage

Similar to the VRP-stage of the GA, the VRP-stage of the ACO algorithm assigns stops to vehicles and optimizes the solutions. The assignment of stops to vehicles and its optimization works with two pheromone matrices as illustrated in Figure 8.4, where each ant represents one vehicle. The *vehicle-to-stop* matrix represents the occupied capacity of vehicles so that ants select the vehicles with enough free space first for representation. The algorithm updates this matrix after each assignment with the current available space of the according vehicle. We performed preliminary tests using a single pheromone matrix which showed us that this value is not enough to determine a good order to vehicle distribution. Instead, the stops that are already assigned to a vehicle have further influence on the final solution as a good clustering of stops per vehicle seems to be advantageous. Hence, we introduce the *stop-to-stop* matrix that covers the distance of stops to stops and is used to determine the next stop to be assigned to a vehicle. By implementing the second matrix, stops with a close distance to each other are more likely to be assigned to the same vehicle: First, an ant selects a stop based on stop-to-stop matrix that is reachable from its current location and has the shortest distance. Then, the ant searches for vehicles that have enough space for this order. We then assign a probability of selecting each of these vehicles by adding the vehicle-to-stop pheromone value (available space) and the stop-to-stop pheromones to all already assigned stops of this vehicle (stop-to-stop distances). Based on these probabilities, the ant selects a vehicle randomly. This means, the higher the amount of aggregated pheromones, the better the vehicle fits for this order, and thus, the higher the probability to select this vehicle.

Algorithm 6 summarizes the behavior of the ACO algorithm using the two pheromone matrices in the VRP-stage. First, the pheromone matrices are initialized with the a priori knowledge of vehicle capacities and stop-to-stop distances (line 1). Additionally, an empty set of solutions is initialized in which

Vehicle-to-Stop Matrix

	Veh. 1	Veh. 2
Stop s1	0	1
Stop s2	1	0
Stop s3	2	3
Stop s4	3	2



Stop-to-Stop Matrix

	s1	s2	s3	s4
s1	1	2	2	1
s2	2	0	1	2
s3	2	1	0	2
s4	1	2	2	1

Figure 8.4: Graph representation of the VRP problem for the ACO algorithm. The representation contains a vehicle-to-stop matrix that represents occupied capacity of vehicles, and the stop-to-stop matrix that covers the distance between stops. The illustrated graph only depicts possible paths and the stop-to-stop matrix further defines the weights of each edge.

the best solutions are stored. The number of stored solutions is defined as twice the number of vehicles of a specific problem instance. We decided to double the vehicle number to have at least one ant per vehicle and a second ant for a further optimization round. Then, a loop starts iterating until a maximum number of iterations that were not able to improve the solution quality are executed (line 2). In each iteration, one ant is placed at the graph and assigns all stops to the vehicles with regards to both pheromone matrices (line 3). In order to keep the idea of Novelty Search [LS08] and avoid getting stuck in local optima, a small amount of distributions are created probabilistic. Afterwards, the algorithm passes a TSP instance per vehicle to the TSP-stage of the ACO which optimizes its sequence (lines 4 and 5). The returned TSP instances are then passed to the Timeline algorithm to match time windows (line 6). Afterwards, the algorithm calculates the final scores for this solution (line 7). Then, the algorithm updates the pheromone matrices using the scores of the solutions in the set and performs a pheromone evaporation step with a probability of 5% which we identified in a preliminary parameter study (lines 9 and 10). If the found solution is better than the worst one in the solution set, or the solution set is not yet full, the solution is added to this set (lines 11 to 14). If the solution is better than the best solution so far, the number of unimproved iterations is reset to zero, else it is incremented. Afterward, the next iteration starts, another ant is placed at the graph and assigns stops to vehicles.

The matrix update in this stage works with a comparison of the score value to the last best and worst scores. Due to the fact, that the algorithm deals with a multi-level priority score, that is any broken constraint in level i is more important than any improvement in level $i + 1$, we decided to include this knowledge in the pheromone update strategy. This way, we want to provide

Algorithm 6: VRP-stage pseudo-code of the ACO.

```
Input: Stops $s$, Vehicles $v$, max. iterations $mi$
1  Initialize pheromone matrices
2  while unimproved iterations $(ui) < mi$ AND current runtime < max. runtime do
3      Assign $s$ to $v$ based on matrices
4      foreach $veh$ in $v$ do
5          Solve TSP-stage
6      Apply Timeline algorithm to match time windows
7      Calculate score of this solution
8      Try add it to set of best solutions
9      Update pheromones
10     Evaporate pheromones
11     if current solution better than best solution then
12         $ui = 0$
13     else
14         $ui++$
15 return best solutions
```

more weight for higher score levels than to lower score levels and direct the search of the algorithm to improve the convergence speed. As summarized in Equation (8.7), the new pheromones for every score level i are calculated by multiplying the score factor f_i with a pheromone base value p_i, divided by the score level (one for H_1, two for H_2, and so on) to give more weight to the more important scores. We distinguish two cases to set p_i: if the current score is better than the worst score ever found, we set $p_i = 1$; if the current score is worse than the worst score, we set $p_i = 0.25$. By this, we give the pheromones of reasonable solutions more weight than of bad ones and hope to gain a faster improvement of the found solutions since many more non-feasible solutions exist. The already mentioned idea of integrating Novelty Search brings the possibility of worse solutions than the currently worst one.

$$\text{pheromones} = \sum_i \frac{f_i \cdot p_i}{i} \tag{8.7}$$

Equation (8.8) shows the calculation of the score factor f_i. The variable ws_i refers to the current worst score, bs_i to the current best score, and s_i to the current score value of the respective level i. By using this formula, we decrease the pheromone amount of solutions with lower scores than the current worst score and exponentially award better solutions.

$$f_i = \left| \frac{(ws_i - s_i)^3}{(ws_i - bs_i)^3} \right| \tag{8.8}$$

8.4.2 TSP-stage

The TSP-stage works with a single *stop-to-stop* pheromone matrix as depicted on the right side of Figure 8.4 representing the probabilities, that is the distance to move from one stop to another. The diagonal values refer to the probability of a stop to be the first stop taking the vehicle's start locations into account. The other values represent the probabilities to move from one stop to another. We initialized this matrix again with knowledge about the stop-to-stop distances and hence, represent the actual distance between the stops from the first iteration onward instead of an equal initialization which would require some time to converge to the actual distances. However, it might happen that order dependencies, order restrictions, or time windows require another stop sequence than shortest first, so we decided to maintain a small probability for every stop. The algorithm starts iterating and places one ant at any location in the graph in every iteration. The ant then decides—depending on the column for the current stop containing the values to every other stop—which stop to visit next. We add a visibility feature to the matrix to guide the ant in a way to first select the pickup stop and afterwards the delivery stop. Hence, we set the visibility of a delivery stop to false if the ant did not pickup the products for this order beforehand and the ant cannot see this stop. This aims at further reducing the convergence time of the algorithm. After one ant finished its walk and returned with a sequence of stops, the algorithm calculates the score for this sequence. Afterwards, the algorithm updates the pheromones similar to the update procedure in the VRP-stage and evaporates the pheromones with a probability of 5%. Further, we apply the principle of Elitism, i.e., the matrix is additionally updated with the current and global best solutions so far, to improve the solution quality even more (cf. [Çat09]). This behavior guides the algorithm to search for better solutions in the neighborhood of already good solutions. The TSP-stage iterates until a maximum number of unimproved iterations occurred, the maximum runtime is exceeded or the path of the ants converged, that is, all ants select the same path.

8.5 Summary

In this chapter, we proposed our main contributions focusing on **Goal B**: *Improving the quality of optimization strategies in complex systems-of-systems with a special attention to the field of logistics*. We answer **RQ B.1** and its subordinate questions **RQ B.1.1** and **RQ B.1.2** by proposing an optimization workflow to deal with the rVRP, which we consider a vertical system-of-systems due to its nested optimization tasks. Besides executing the proposed approaches once, they can also be integrated into a feedback loop to adapt the solution to a changing environment. Therefore, in terms of the terminology of the proposed framework from Chapter 7, we consider the proposed optimization approaches as adaptation planning strategies. With the knowledge of this kind of vertical systems-of-systems, the proposed framework can be extended to cope with such complex systems and be applicable to a wider range of use cases.

Chapter 9

Optimization of Horizontal Systems-of-Systems

The previous chapter proposed approaches for a kind of vertical systems-of-systems [Mai98] where the overall problem statement can be divided into two subsystems handling nested problem statements while still remaining functional as individual components. In the following, we address another kind of systems-of-systems consisting of two components that coexist and cooperate next to each other and influence the output quality of each other. The use case of this chapter is part of the logistics domain and handles the optimization of mezzanine warehouses. Working within a mezzanine warehouse consists of two main tasks: (i) filling the storage with goods (storage assignment) and (ii) picking items out of the storage (order picking). The storage assignment problem defines the task of selecting storage locations to put a product into storage. On the contrary, the order picking problem handles the task of computing a pick route that collects the requested products of a customer order. Finding suitable storage allocations is important, as the allocation of products affects the travel distances during order picking. Thus, optimizing each warehouse problem individually may yield suboptimal solutions, harming the overall warehouse performance. Hence, these warehouse problems coexist and influence each other in parallel [GGM10] and, thus, we consider this problem statement to be a horizontal system-of-system. We address the optimization warehouses in this section as prototypical use case for a complex system-of-systems in the logistics domain. Still, the contribution of this chapter can be generalized to other domains such as smart grid or intelligent computer networks.

In this chapter, we propose an integrated approach for combined storage assignment and order picking that simultaneously optimizes multiple economic and ergonomic constraints in mezzanine warehouses. Regarding the already presented layered architecture in Section 7.3, we consider the proposed approaches of this chapter as part of the adaptation planning layer. Besides the single execution of the proposed approaches for each problem statement, the approaches can also be applied regularly using a feedback loop to adapt the solution to a possibly changing environment. In this chapter, we aim at handling the effects of optimization of horizontal systems-of-systems. However, the actual adaptation of our proposed framework introduced in Chapter 7 is out of scope of this thesis. Still, the lessons learned from this chapter will be useful for adapting the framework to handle not only a single adaptive system but also systems-of-systems. Hence, in this chapter, we apply optimization techniques on the parallel warehouse problem statements order picking and storage assignment with a special focus on their interaction. To solve these problems, we integrate economic as well as ergonomic criteria. Expert interviews have shown that in practice, the following set of economic criteria is important and, hence, supported by our approach: products should be spread equally among each floor, fast-moving products should be easily accessible, correlated products should be stored in proximity of each other, and the storage space should be used as efficiently as possible. Further, we integrate ergonomic constraints such as storing heavy products and fast-moving products at grip height or reducing the requirement to switch a mezzanine floor. Hence, the contributions of this chapter are two-fold.

- Design of storage allocation and order picking algorithms that incorporate the interdependence of both tasks.
- Integration of diverse economic and ergonomic constraints.

This chapter is based on our paper [LMK+21a, LMK+21b], which is currently under review in the Springer Applied Intelligence Journal. In the following, we first propose our meta-model of considered mezzanine warehouses in Section 9.1. Afterwards, we provide an overview of the goal and a 3-phase algorithm of our storage assignment approach in Section 9.2 and present the details of the proposed GA in Section 9.3. Finally, we introduce our order picking approach based on an adapted ACO algorithm in Section 9.4 and summarize the chapter in Section 9.5.

9.1 Meta-Model of Considered Mezzanine Warehouses

The storage assignment and order picking algorithms require information on the warehouse layout, the product assortment, the products' storage locations, and the current state of the warehouse. This state represents the already assigned products and their location. Figure 9.1 illustrates our meta-model.

The mezzanine warehouse consists of multiple floors but we consider only one floor in the meta-model. To model multiple floors multiple instances of

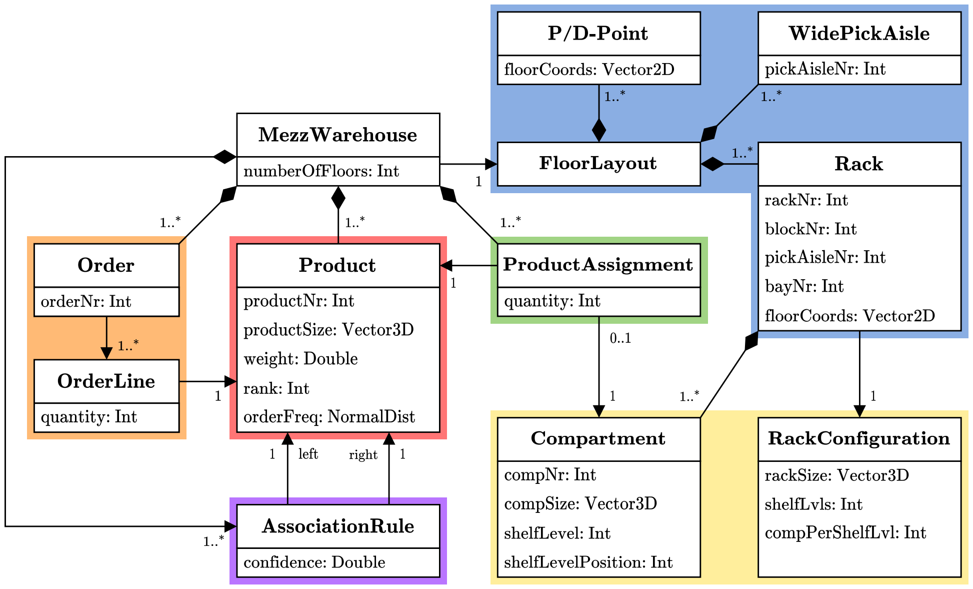


Figure 9.1: The meta-model describes the structure and state of mezzanine warehouses. It consists of information about the floor layout (blue classes), configuration of a rack (yellow classes), the contained products (red class), the product assignment (green class), a set of orders (orange classes), and a set of association rules for the products (purple class).

the presented model can be combined. The blue box describes the floor layout defining the arrangement of racks within one floor of the mezzanine warehouse (`FloorLayout`). Each floor consists of the classes, `P&D-Point`, `WidePick-Aisle`, and `Rack`. A p/d-point is the pickup and delivery point where personnel needs to collect items to be stored in the warehouse or deliver items of a customer order that were collected. Regarding the class `WidePickAisle`, two types of pick aisles exist: wide and narrow ones. In wide aisles, pickers can take along their cart to cross the aisle while it needs to be parked at the aisle entry for narrow aisles. A floor can be illustrated as a two-dimensional map as depicted in Figure 9.2: The racks with their unique identifiers r_3 and r_4 are assigned the floor coordinates $x = 1$ and $y = 2$ since their access points are both located at $(1|2)$. The vertical aisles located at $x = 0$ and $x = 4$ in combination with the horizontal cross aisles at $y = 0$, $y = 4$, and $y = 7$ form the periphery of the floor. Periphery aisles usually contain the p/d-points, e.g., at $(2|0)$). A wide aisle is depicted at x-coordinate two and two narrow aisles are shown at x-coordinates one and three, where the picker needs to park his cart. Real-world mezzanine warehouses may apply different layouts on each floor; however, we assume that each floor in the mezzanine warehouse applies the same layout.

Since diagonal movements are not possible in this grid layout, we apply the Manhattan distance function to calculate the distance between two locations p and q defined by $distance(p, q) =\mid x_p - x_q \mid + \mid y_p - y_q \mid$.

The classes inside the yellow box (`Compartment` and `RackConfiguration`) define the configuration of a rack, referring to its size, the number of shelf levels, the number of compartments per shelf level, and the size of the compartmens. The `Compartment` class includes an identifier and a three-dimensional vector specifying the compartment's dimensions. The shelf level and shelf level position define the compartment's location within the rack. The class `Product` (red box) defines the products with five properties: product number, size, weight, rank, and order frequency. The rank (≥ 1) allows identifying fast and slow-moving products by the frequency at which the product appears in recent customer orders. The product of rank 1 represents the most frequently ordered product. The order frequency describes the frequency to which a product is usually ordered using a Gaussian distribution. Both properties are derived from recent customer orders and represent redundant information which prevents the algorithms from recalculating this information in every computation step it is needed. Further, these properties are later used in the storage assignment optimization to find better racks regarding their frequency and usual ordered amount. The class `ProductAssignment` (green box) specifies the quantity of which a product is assigned to a specific compartment. The classes `Order` and `OrderLine` of the orange package define the structure of a customer order consisting of a unique order number and multiple order lines. An order line specifies the quantity to which a product is ordered. The class `AssociationRule` (purple box) defines association rules derived by the Apriori algorithm [ICMPG16]. The confidence ranges from 0 to 1 and expresses the strength of the correlation between the left-sided and the right-sided set of products. In this thesis, we limit our approach on positive correlations but the integration of negative correlations can also be realized with minimum overhead in the assignment process. These rules are used in the storage assignment algorithm later on to store correlated products close to each other which may increase the order picking performance.

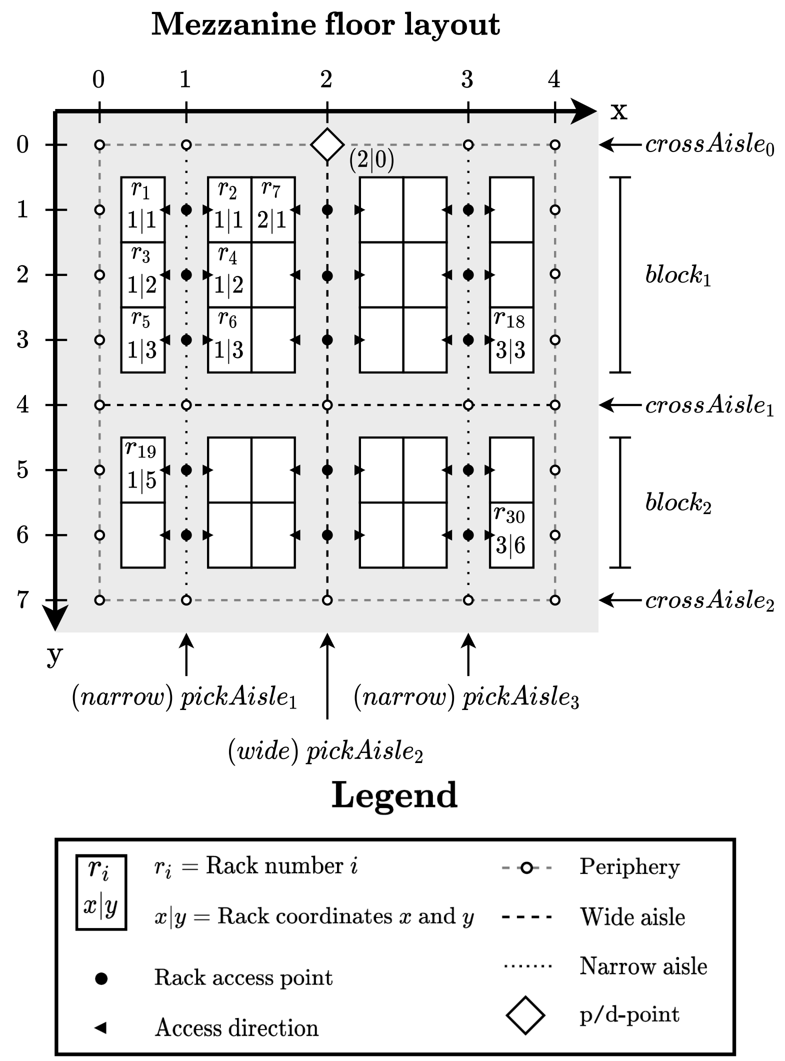


Figure 9.2: Example mezzanine floor layout from top-down view. We structure the floor layout using two-dimensional coordinates and consider narrow (at $x = 1$ and $x = 3$) and wide pick aisles (at $x = 2$) as well as cross aisle (at $y = 0, 4, 6$). Further we show P&D-points for example at $(2|0)$. We assign rack ids to the shown racks and define rack access points from the aisle.

9.2 Storage Assignment

The overall goal of the storage assignment algorithm is to select a set of compartments for storing an incoming product by considering multiple economic and ergonomic constraints simultaneously. In the following, we first introduce the defined constraints and assumptions for the storage assignment approach. Afterwards, we present the general structure of the storage assignment algorithm consisting of three phases.

9.2.1 Constraints and Assumptions

In expert interviews, we identified multiple hard constraints that should be covered in the proposed approaches. These hard constraints specify whether a storage allocation is considered feasible. Such a feasible solution never violates any of the following constraints. Each incoming item must be assigned to a compartment (HC_1). The selected compartment must either be empty or partially occupied by items of the same product (HC_2). Each item has to fit in the remaining free space of its assigned compartment (HC_3). Furthermore, we define multiple soft constraints that measure the extent to which a storage allocation fulfills economic criteria: The products should be evenly spread on each floor (SC_1). Fast-moving products should be assigned close to a P&D-point (SC_2). The mean ordered quantity of a product should be locally available (SC_3). Correlated products should be stored close to each other (SC_4). The storage space should be used as efficiently as possible (SC_5). Finally, we define two ergonomic soft constraints: Heavy products should be stored at grip height (SC_6). Fast-moving products should be assigned at grip height (SC_7).

Further, we state the following assumptions for our approach: The state of the warehouse does not change while the storage assignment algorithm is running. Thus, the products are neither re-positioned nor removed, and the racks' configurations do not change. Further, the algorithm allocates only one product at a time. The storage racks may apply different rack configurations and products may only be assigned to fitting compartments. A compartment is allowed to store multiple items of the same product but may not store two different products at the same time. Finally, we assume that one execution of the storage assignment plans locations for a single product type and an assignment task for other product types requires an additional run of the algorithm.

9.2.2 3-Phase Storage Assignment Algorithm

Our storage assignment algorithm consists of three phases that intend to reduce the complexity of the optimization problem: (i) assignment of products to floors, (ii) assignment to racks w.r.t. economic criteria, and (iii) assignment to compartments w.r.t. ergonomic criteria. Keep in mind that we only consider feasible solutions which fulfill all hard constraints.

In the first phase, the incoming product quantity is split among the mezzanine floors (SC_1) so that each floor provides the same quantity of the product. This way, we try to reduce the required floor changes during a pick route to a minimum. Thus, we first determine the total quantity of the incoming product that is already available in each floor, calculate the ideal quantity for each floor after storage assignment, and assign the missing quantity to each floor. Remaining items, due to rounded results, are allocated to a random floor.

The second phase addresses the economic soft constraints SC_2 to SC_5 and aims to reduce the walking distances during order picking. This phase assigns the incoming products to racks on a specific floor. Since this phase requires optimizing a set of constraints, we apply a multi-objective optimization algorithm which we describe in Section 9.3.

The third phase aims to satisfy the ergonomic soft constraints SC_6 and SC_7. We classify a product p into three weight classes: *light* (up to 3 kg), *medium* (between 3 kg and 7 kg), and *heavy* (over 7 kg). We set the grip height from 0.75 m to 1.25 m and refer to compartments below/above the grip height as low/high zone compartments, respectively. Additionally, we distinguish *fast-moving*, *moderately-moving*, and *slow-moving* products by their relative rank. The relative rank of a product p calculated as $rank_p/|P|$, where $rank_p$ denotes the rank of product p, and $|P|$ the size of the product assortment, that is the set of unique products. In the first step, we assign the incoming items to the rack's compartments that already provide items of the same product. In the second step, we assign the remaining incoming items to the rack's unoccupied compartments based on predefined penalty values. The penalty values range from zero to three and the more a compartment is unsuited for storing the product, the more penalty points are given (see Table 9.1 and Table 9.2).

Table 9.1: Penalties for assigning a product to a specific compartment with regards to the product weight.

Zone \ Weight	heavy	medium	light
high	3	2	0
grip height	0	0	1
low	1	1	0

Table 9.2: Penalties for assigning a product to a specific compartment with regards to the product rank.

Zone \ Rank	fast	moderate	slow
high	2	0	0
grip height	0	1	3
low	2	0	0

9.3 Genetic Algorithm for Storage Assignment

This section presents our custom version of the NSGA-II algorithm that was originally proposed by [DPAM02]. We select the GA as it is a nature-inspired heuristic algorithm which is commonly used to tackle such complex problems. Still, also other heuristic optimization techniques could be applied in the future to assess their performance. The algorithm receives the current state of a floor, that is, instances of the meta-model for the addressed floor, and assigns the incoming items to a set of racks on this floor. Note that we execute the NSGA-II once for each floor and, hence, solve the optimization problem for each floor individually considering economic constraints.

9.3.1 Chromosome Encoding

Since the NSGA-II algorithm is a genetic algorithm, we propose to use a customized chromosome encoding as depicted in Figure 9.3. The figure illustrates an example allocation task where ten items of product p_1 must be assigned to the racks on f_1. Black numbers written within the racks indicate existing items of product p_1, while red numbers indicate incoming items of product p_1. The right side shows the chromosome that encodes the storage allocation depicted on the left side by specifying the racks selected for storing each incoming item. Since ten items of product p_1 are assigned, the chromosome's length equals 10.

9.3.2 Objective Functions

A set of objective functions guides the NSGA-II algorithm to find good storage allocations. We propose four domain specific objective functions, which we

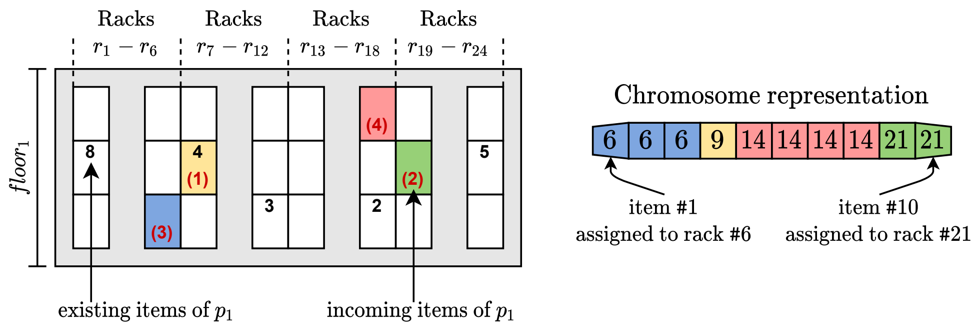


Figure 9.3: Illustration of the chromosome representation and the reference to the warehouse layout. The chromosome on the right encodes the racks selected for storing the incoming items inside the warehouse depicted on the left. Black numbers indicate existing items of a product, while red numbers indicate incoming items of product.

call scores, for our maximization problem: (i) spread score, (ii) distance score, (iii) quantity score, (iv) correlation score.

9.3.2.1 Spread Score

This score addresses constraint SC_1 and aims to equally spread the incoming quantity of product p_1 across the entire floor. Hence, we divide the floor f_j into multiple areas A of equal size. To calculate the spread score, we use the total $(totalQ)$ and ideal quantity $(idealQ)$ of a product in each area of a floor. The $totalQ$ is the sum of the existing and incoming items in an area, while we calculate the $idealQ$ by dividing the sum of the existing and incoming quantity of product p_1 on the floor by the number of defined areas. The aim of this calculation is to balance the amount of products in all areas and, hence, achieve a good spread of products within the entire floor. We define the final spread score for chromosome C to be the sum of differences between the total and the ideal quantity for all areas multiplied with minus one (see Equation 9.1).

$$spreadScore_{p_1,f_j} = (-1)\sum_{o=1}^{|A|}|idealQ_{p_1,f_j,o} - totalQ_{p_1,f_j,o}| \qquad (9.1)$$

9.3.2.2 Distance Score

This score addresses constraint SC_2 and aims to allocate slow-moving products to racks further away from the P&D-points. Hence, the distance score quantifies the extent to which the walking distances $(dist_{r_i})$ of the selected racks match the ideal distance ($idealDist_{p,f_j}$). For calculating the $idealDist$, we perform the following steps: We determine the relative rank of the incoming product p by dividing the rank of the product ($rank_p$) by the size of the product assortment P: $relRank_p = rank_p/|P|$. We map the relative rank to a rack index:

$$rackIdx_{p,f_j} = \lfloor relRank_p \cdot |R_{f_j}| \rfloor$$

with R_{f_j} being the list of racks of floor f_j sorted by the racks' walking distances to their closest P&D-point. The rack in R_{f_j} at index $rackIdx$ represents the best-suited rack for storing product p with regard to constraint SC_2. Finally, we define the $idealDist$ to be:

$$idealDist_{p,f_j} = distance(R_{f_j}[rackIdx_{p,f_j}]) \tag{9.2}$$

The overall distance score is calculated as the sum over all racks in Chromosome (C) of absolute differences between the walking distances of the racks selected for storing product p and the $idealDist$ (see Equation 9.3).

$$distanceScore_{p,f_j,C} = (-1)\sum_{n \in C} |idealDist_{p,f_j} - dist_n| \tag{9.3}$$

Further, we provide an example of the procedure to calculate the optimum walking distance in Figure 9.4. The left side depicts the floor layout with already existing products within racks of the floor depicted as black number and black points representing P&D-points. First of all, we classify all racks of the floor with regards to their walking distance to the next P&D-point, which results in eight, twelve, twelve, and four racks with a walking distance of one, two, three, and four, respectively. This classification of racks is also depicted at the top of the right illustration. Now, we analyze the rank of the product to be stored and calculate the relative rank. As an example, we assume the product to have a rank of 98 and a product assortment size of 160 what results in a relative rank of 0.61 for the product to be stored. Then, we map this rank to the rack indices that can be illustrated as an ordered list of racks. In our example, as we assume to have 36 racks at the floor, the relative rank of 0.61 maps to a rack index 23, which again maps to the ordered list of classified racks with regards to their walking distance. Since we identified the optimum rack index to be 23, we select the 23rd rack in the classified ordered list. The selected rack is classified as a rack with walking distance three, which defines the optimum walking distance for this product as three. This value is then used in the above mentioned calculation to identify the distance score.

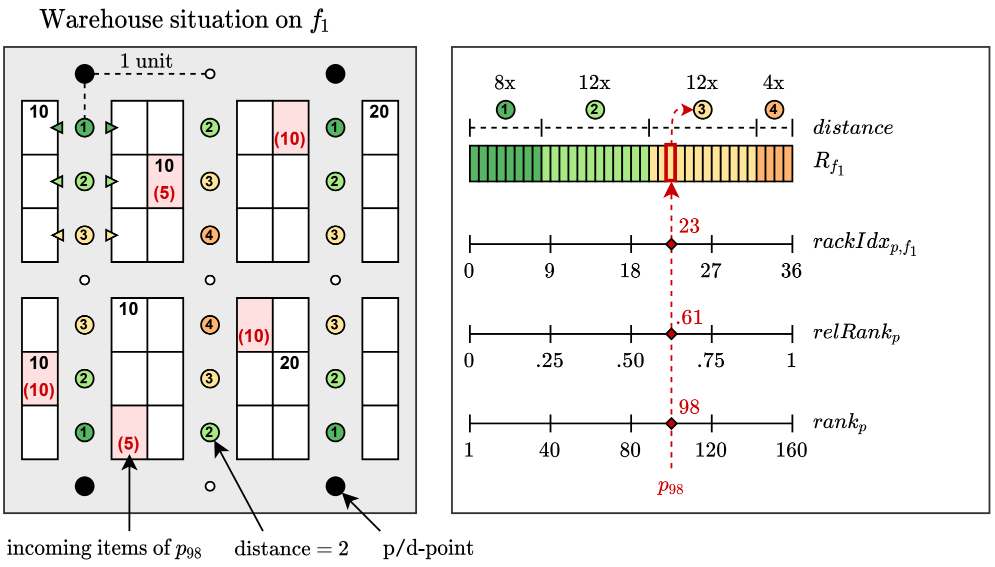


Figure 9.4: Calculation of the ideal distance for storing the incoming product p_{98}. The left side illustrates the floor layout with P&D-points, 36 racks, the walking distances for each rack and already assigned products as black numbers inside the racks. The right side illustrates the process to determine the optimum walking distance for a product regarding its rank.

9.3.2.3 Quantity Score

This score assesses SC_3 and ensures that the mean ordered quantity of a product is locally available. With the term locally, we refer to a predefined area within a floor, which we specify in the following using four masks. Therefore, the target quantity defines the quantity to which the product p should be locally available based on a set of recent customer orders calculated using mean (μ) and standard deviation (σ): $tq_p = \lceil \mu_p + 2\sigma_p \rceil$. In the following, we refer to a pick aisle within one block in a floor as sub aisle. Further, we define four masks and a modifier for each mask to measure the density to which the tq_p is locally available: M_1 equals the size of a rack ($maskMod = 1$), M_2 equals the size of two facing racks ($maskMod = 0.75$), M_3 is a sliding window with half the sub aisle's length ($maskMod = 0.5$), and M_4 covers an entire sub aisle ($maskMod = 0.25$). We define the mask modifier in the range between one and zero with decreasing values to give less impact of smaller masks to the mask score. In this thesis, we use these predefined values but they can be tuned in future work to find the optimal configuration. Using these masks, we calculate a quantity factor for each sub aisle (sa) of a floor and each mask (M_k). Therefore, we select the quantity (q) of products inside a mask divided by the target quantity:

$$qFactor_{p,f_j,sa_l}(M_k) = \max_{M_k}(q_{p,f_j,sa_l}(M_k)/tq_p)$$

This results in a value of one or higher if the target quantity is met and a value of zero if no products can be found within this mask. We then multiply this quantity factor by the $maskMod$ to calculate the mask score:

$$maskScore_{p,f_j,sa_l}(M_k) = \min(maskMod_{M_k} \cdot qFactor_{p,f_j,sa_l}(M_k), 1)$$

We define the mas score in the range between zero and one with a value of one indicating that the target quantity is available in a single rack of the sub aisle. From these we select the maximum value to assign a score to each sub aisle:

$$subAisleScore_{p,f_j,sa_l} = \max_{M} maskScore_{p,f_j,sa_l}(M_k)$$

The final quantity score computes as the sum of all $subAisleScore$s:

$$quantityScore_{p,f_j,C} = \sum_{o=1}^{|SA|} subAisleScore_{p,f_j,sa_o} \tag{9.4}$$

Figure 9.5 illustrates the idea of using *masks* of different sizes to measure the density to which the target quantity tq_p of product p is locally available. The left side shows the storage locations of existing and incoming items of product p in a specific sub aisle sa. The center of the figure depicts the four masks M_k that iterate over the racks of the sub aisle. During this process, the masks count the existing and incoming quantities of product p that can be found in the covered regions. The right side shows the regions where the masks find the largest quantity of product p in the sub aisle sa.

9.3.2.4 Correlation Score

This score relates to SC_4 and describes the extent to which the incoming product is stored close to its correlated products. Association rules describe correlations between products and can be derived from recent customer orders. We consider association rules of the form $rule = \{p\} \xrightarrow{conf} \{cp\}$, where p denotes the incoming product, cp the correlated product, and $conf$ a confidence value.

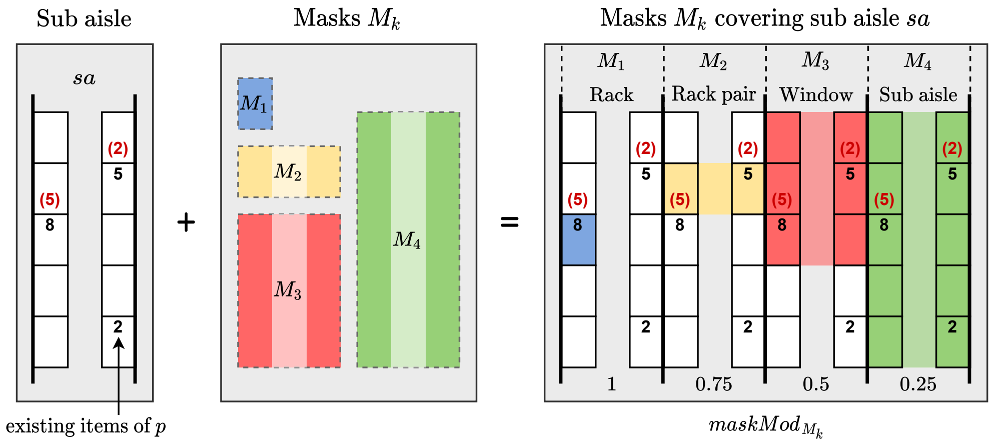


Figure 9.5: Illustration of the workflow for calculating the quantity score. The left side depicts a sub aisle layout with existing and incoming items. The center of the figure depicts the defined masks. The masks M_k count the quantities of product p in the covered regions which results in the right of the figure where the masks find the regions with the largest quantity of a product.

We first calculate the number of possible clusters of target quantities of the incoming product:

$$qClusters_{p,f_j} = \lfloor totalQ_{p,f_j} / tq_p \rfloor$$

We use this value to define the ideal quantity to which the correlated product should be available in the vicinity of the incoming product:

$$idealCorrQ_{rule,f_j} = \lceil qClusters_{p,f_j} \cdot tq_{cp} \cdot conf(rule) \rceil$$

In the next step, we determine the quantity of cp already available in the vicinity of p. For this task, we use the previously introduced masks M_k and place them directly on top of the racks containing cp. We again calculate the $qFactor$ to capture the extent to which the target quantity of p is available in the region covered by M_k placed on top of rack r:

$$qFactor_{p,r}(M_k) = q_{p,r}(M_k) / tq_p$$

Then, we calculate the fraction to which the items of cp stored in r are considered to be in the vicinity of p:

$$corrQ_{rule,r}(M_k) = exQ_{cp,r} \cdot qFactor_{p,r}(M_k) \cdot maskMod_{M_k}$$

$exQ_{cp,r}$ refers to the existing quantity of the correlated product cp in rack r. Afterward, we select the $corrQ$ with the highest value representing the mask with the largest amount of p in the vicinity of cp:

$$corrQ_{rule,r} = \max_{M} \ corrQ_{rule,r}(M_k)$$

The sum of all $corrQ_{rule,r}$ over all racks on this floor denotes the quantity of the cp on this floor that is considered as being in the vicinity of p:

$$corrQ_{rule,f_j} = \sum_{r \in R_{f_j,cp}} corrQ_{rule,r}$$

Now, we calculated the quantity of the correlated product that is in the vicinity of the incoming product and the difference of this value to the ideal quantity. Based on this difference, the correlation score is calculated as:

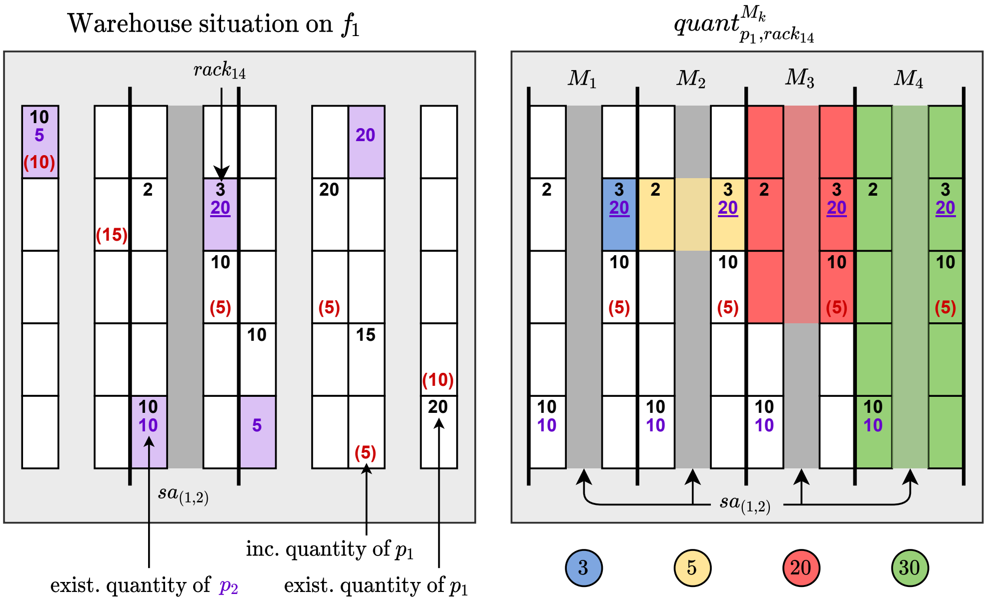
$$correlationScore_{p,f_j,C} = (-1) \sum_{rule \in A_p} idealCorrQ_{rule,f_j} - corrQ_{rule,f_j} \quad (9.5)$$

The $correlationScore_{p,f_j,C}$ is the correlation score awarded to a storage allocation encoded as chromosome C that assigns items of product p to racks on f_j. A_p represents the set of association rules that have the incoming product p on the left side.

Again, we illustrate the calculation of the correlation quantity for the correlation score in Figure 9.6. The top of the figure presents given information required for the calculation, such as existing and incoming quantities for the example product p_1 as well as the existing quantity of product p_2. Further, one association rule and two target quantities are specified. The left box of the figure illustrates the warehouse situation on the example floor f_1 where existing quantity of p_1 is written in black numbers, incoming quantity of this product is written in red, and existing quantity of p_2 is indicated using the purple color. The right box of the figure illustrates the mask placement on the existing quantity of p_1, and counts the existing quantity of p_1 inside of the mask in the colored circles below this illustration. The table at the bottom of the figure illustrates the calculation of the correlation quantity for each mask. For example, considering mask M_1 depicted in blue, the correlation quantity is calculated by multiplying the existing quantity of p_2 (= 20) by the $qFactor$, that is, existing quantity of p_1 divided by the target quantity which results in $\frac{3}{10}$. Finally, we multiply this by the mask modifier of mask M_1, which we define to be one. We apply this process to all masks and select the maximum correlation quantity for the further calculation of the correlation score.

Given:

- $exQ_{p_1,f_1} = 100$, and $inQ_{p_1,f_1} = 50$, and $exQ_{p_2,floor_1} = 60$
- $A_{p_1} = \{rule_1\}$, where $rule_1 = \{p_1\} \rightarrow \{p_2\}$, with $conf(rule_1) = 0.7$
- $tq_{p_1} = 10$, and $tq_{p_2} = 5$



$corrQ^{M_1}_{rule_1,rack_{14}}$		$corrQ^{M_2}_{rule_1,rack_{14}}$		$corrQ^{M_3}_{rule_1,rack_{14}}$		$corrQ^{M_4}_{rule_1,rack_{14}}$	
3	$20 \cdot \frac{3}{10} \cdot 1 = 6$	5	$20 \cdot \frac{5}{10} \cdot 0.75 = 7.5$	20	$20 \cdot \frac{10}{10} \cdot 0.5 = \underline{10}$	30	$20 \cdot \frac{10}{10} \cdot 0.25 = 5$

Figure 9.6: Illustration of the calculation of the correlated product p_2 available in the vicinity of the incoming product p_1. The left side depicts the warehouse situation on the first floor with existing and incoming quantities of p_1 as well as existing quantities of p_2. The right side depicts the placed masks on this sub aisle and the correlation quantity calculation is depicted at the bottom.

9.3.3 Genetic Operators

The NSGA-II is a generally applicable genetic algorithm but provides the possibility to define use case specific selection, crossover, and mutation operators. A customized use case specific definition of these operators might help the NSGA-II to find better solutions in a smaller amount of iterations and, thus, decreases the convergence speed. Hence, these operators need to be defined carefully so that the algorithm constructs solutions that achieve good objective function values. We apply selection and crossover operators that are well-established in the literature and focus on use case specific mutation operators.

Selection. We apply a binary tournament selection operator where two random parent individuals compete against each other [DPAM02]. The individual with the higher Pareto rank is the winner and is allowed to participate in the crossover procedure. In case both parents are of equal Pareto rank, the individual with the larger crowding distance, i.e., the higher diversity, wins the round. We apply these rounds until only two individuals remain in the finals of the tournament. On these two individuals, we breed new individuals by applying the crossover operator.

Crossover. Since all chromosomes created during a run of the NSGA-II algorithm are of equal length, we use the traditional single-point crossover operator. It selects a random crossover point on both parents' chromosomes, splits them, and recombines them cross-wise to obtain two new children.

Mutation. We define the following mutation operators that incorporate domain-specific knowledge to guide the search process and achieve faster convergence speed while maintaining a high diversity of the population:

1. The **FillRack** mutator selects a random rack and fills it with incoming items from the same sub aisle.
2. The **MoveRack** mutator selects a random rack containing at least one incoming item and moves them to a different rack within the same sub aisle.
3. The **FillSubAisle** mutator selects a random sub aisle and fills it with incoming items from other sub aisles until it provides the product's target quantity.
4. The **ClearSubAisle** mutator selects a random sub aisle and moves any incoming items to a different sub aisle.
5. The **RedistributeExceedingQuantities** mutator redistributes incoming items of racks that provide more items than the target quantity to racks that require only a few items to provide the target quantity.
6. The **ShiftRacks** mutator shifts all incoming items towards a randomly selected direction: left, right, up, or down.
7. The **SwapSubAisles** mutator first groups the sub aisles into pairs and swaps incoming items randomly within each pair.

8. The **SwapRacks** mutator is similar to (7) but swaps items within pairs of racks instead of sub aisles.

9.3.4 NSGA-II Algorithm

We summarize the overall procedure of our NSGA-II algorithm in Algorithm 7. The algorithm receives the `product` to be stored and its `quantity` as well as the

Algorithm 7: Proposed NSGA-II Algorithm.

```
Input: product, quantity, fittingRacks
Parameter: parentPopSize, mutProb, $L$, $\delta_{lim}$, maxGen
Output: paretoFront
pop$_{parent}$ = initParentPopulation(product, quantity, fittingRacks,
  parentPopSize)
gen = 0
historyOfMaxCD = new List()
while gen < maxGen && $std(L) > \delta_{lim}$ do
    gen++
    pop$_{children}$ = createChildrenPopulation(pop$_{parent}$)
    pop$_{combined}$ = pop$_{parent}$ ∪ pop$_{children}$
    pop$_{parent}$ = createNextParentPopulation(pop$_{combined}$, parentPopSize)
    paretoFront = calculateParetoFront(pop$_{parent}$)
    maxCD = calculateMaxCD(paretoFront)
    historyOfMaxCD.add(maxCD)
return calculateParetoFront(pop$_{parent}$)
```

list `fittingRacks` as input parameters. Further, the `parentPopSize` defines the size of the parent population, the mutation probability is given by `mutProb`, the number of generations to be used when calculating the standard deviation of the maximum crowding distance $std(L)$ is called L, the threshold for the standard deviation of the crowding distance is δ_{lim}, and the maximum number of generations is called `maxGen`. In the end, the algorithm returns a `paretoFront` of the best storage assignments.

In the first step, the algorithm initializes the population by randomly creating the required amount of chromosomes in line 1. Therefore, the algorithm selects fitting racks for the product randomly which might produce invalid solutions due to exceeded rack spaces. Each invalid chromosome is then repaired by moving the amount of exceeding products to another available rack. Then, the generation counter `gen` in line 2 and the history of observed maximum crowding distances in line 3 are initialized. Afterwards, the while loop starts and iterates using the two following stopping criteria in line 4: (i) the number of maximum generations (`maxGen`) is executed, or (ii) the standard deviation of observed crowding distances ($std(L)$) falls below the given threshold (δ_{lim}). Inside the while loop, the algorithm increments the generations counter (line 5), and breeds a complete new children population in the size of the parent population using the proposed selection, crossover, and mutation operators (`createChildrenPopulation()` in line 6). The algorithm adds this set to a combined population of existing parent individuals (line 7) and selects the best individuals to fill the new parent population (`createNextParentPopulation()` in line 8). Afterwards, the algorithm calculates a Pareto front from this parent population (`calculateParetoFront()` in line 9) and calculates the maximum crowding distance of this front in line 10. This value is added to the history of maximum crowding distances in line 11. In case the while loop stops, the algorithm returns the current Pareto front.

Since the NSGA-II algorithm returns a Pareto front, a user is usually required to identify the most valuable trade-off solution. However, we automate this step by applying the following procedure. For each of the four objective functions (of_i), we select the solution (s_j) of the Pareto front with the highest value ($of_i(s_j)$) for this function. We then use these values as a 4-dimensional reference point ($p_{ref} = [e_1, e_2, e_3, e_4]$). Based on the Euclidean distance, the solution that is closest to the reference point is automatically selected as the most valuable trade-off solution.

9.4 Order Picking

This section introduces our order picking approach that is based on ACO. We selected this algorithm as it is a nature-inspired heuristic algorithm which is commonly used to tackle these complex problem statements. Still, also other heuristic optimization techniques could be applied in the future to assess their performance. The overall goal of this algorithm is to construct a pick route for a given customer order. Since the travel distance is an essential economic goal, the pick route should be as short as possible. Additionally, the pick route should also be ergonomically favorable. The need for changing floors should be minimal to reduce the order picker's physical stress. Further, the product picking sequence is relevant since if light products are picked first, the order picker might need to rearrange the already picked products so that light products are placed on top of heavy products. Hence, the algorithm aims to construct a short pick route that collects heavy products first and changes floors as little as possible to address economic and ergonomic criteria.

The main idea of this approach is to represent a mezzanine warehouse as a graph and let ants search for satisfactory order picking sequences. We make the following assumptions to better deal with the complexity of the problem:

- The state of the warehouse does not change while the algorithm is running, that is, no re-positioning or removal of products is performed.
- The start and ending P&D points of a pick route may differ.
- Narrow sub aisles may only be traversed to the sub aisles' midpoint, as the picker always must go back to the cart in the wide pick aisle.
- The order pickers visit only one rack each time they enter a sub aisle.
- Picking carts withstand infinite loads and can carry any number of items.

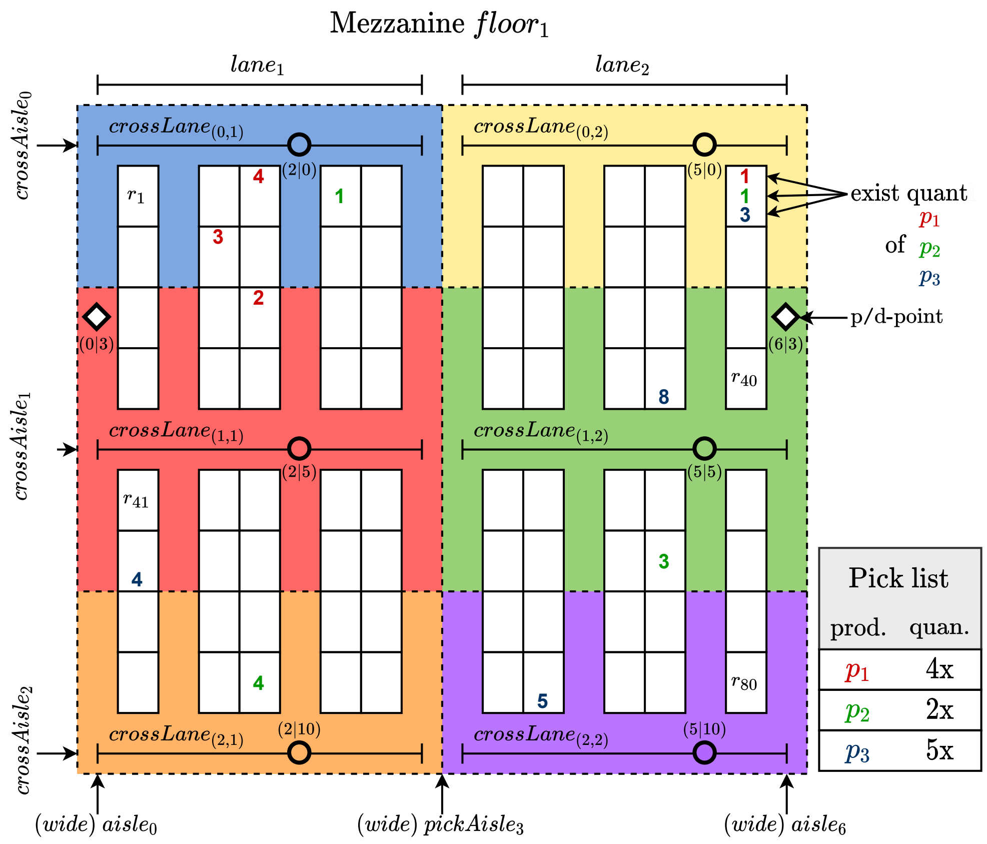
9.4.1 Constraints

For the order picking algorithm, we define a set of hard and soft constraints. The hard constraints assess the feasibility of a solution, while the soft constraints measure the extent to which the solution fulfills economic and ergonomic goals. We define the following hard constraints:

- HC_1: The pick route must start and end at a P&D point.
- HC_2: The pick route must collect the requested quantities of the products specified in the pick list.
- HC_3: After entering a narrow sub aisle, the route must always return to the sub aisle's entrance.

Further, we define economic and ergonomic soft constraints:

- SC_1: The travel distance should be minimal (economic).
- SC_2: The need for changing floors should be minimal (ergonomic).
- SC_3: Heavy products should be picked first, followed by lighter products (ergonomic).



Pick list	
prod.	quan.
p_1	4x
p_2	2x
p_3	5x

Figure 9.7: Illustration of the graph representation for one exemplary floor. The floor is divided into multiple market zones using different colors. Each market refers to one node in the graph that connects to the nearby markets using the travel distances as weights for each edge.

9.4.2 Graph Representation

We propose the following procedure for transferring a mezzanine warehouse into a graph representation as required by the ACO. Figure 9.7 illustrates the procedure of dividing the warehouse into multiple zones called market zones. Each market zone is represented by a market, and thus, a node in the graph. The figure depicts the state of floor f_1 that consists of three cross aisles, three wide (pick) aisles, and two p/d-points. We define six market zones (illustrated using different colored areas) obtained by dividing the floor along the wide (pick) aisles into multiple vertical lanes. In the depicted example, $lane_1$ refers to the area from $aisle_0$ to $pickAisle_3$, and $lane_2$ refers to the area from $pickAisle_3$ to

$aisle_6$. A $crossLane_{(c,l)}$ refers to the part of the cross aisle c that lies within the lane l. For each cross lane, we define a market zone that comprises the storage racks that can be visited from the respective cross lane up to their midpoints. For example, the red market zone includes the racks that can be visited if the order picker is located at the $crossLane_{(1,1)}$. The market zones are limited to the midpoints of the corresponding sub aisles, which prevents the ants from constructing pick routes that entirely traverse the pick aisles. A market is referred to as $market_{(f,c,l)}$, where f denotes the floor, c the cross aisle, and l the lane. For each market, we define three attributes: (1) the market's *coordinates*, (2) the market's *closest p/d-point*, and (3) the market's *supply* that specifies which products are available at which quantity. After defining all markets, they are connected via edges to create a complete directed graph. The edges' weights represent the Manhattan distances between the markets. If the warehouse consists of a second floor f_2, the markets on f_1 are also connected to the markets on f_2 and vice versa, with an extra $floorPenalty$ added to the edges' weights. An adjacency matrix A represents the complete directed graph.

9.4.3 Pick Route Construction

An ant colony explores the created graph to construct a set of pick routes, i.e., a sequence of markets that provide the products, for a given pick list. Hence, a pick route is a path within the graph visiting a series of markets that provide the requested product quantities. A pick route consists of two layers: (i) sequence of visited markets, and (ii) rack sequences. Figure 9.8 depicts an example pick route created by a single ant of the colony. The market sequence (layer one) of a pick route is computed by an ant that is placed on a market within the graph. Guided by the pheromone trails, the ant visits neighboring markets until it collected the requested product quantities specified in the pick list. The ant manages a purchasing list that specifies the missing items. The pick route is complete after the ant's purchasing list is empty. Further, each ant must decide whether it enters the market zone from the left or from the right side. The left/right entrance is located at the position where the cross lane has its lowest/highest x coordinate value. The decision from which side the ant enters the market zone depends on the position of the previously visited market. While constructing pick routes, the ant applies a heuristic function to identify the markets within its vicinity that seem attractive to visit next.

The second layer represents the rack sequence, i.e., the racks the ant visited in each market. We use one ant for both layers, that is, the ant the identifies the market sequence also defines the rack sequence according to the following priority rules: (1) Racks that provide heavy products should be visited first.

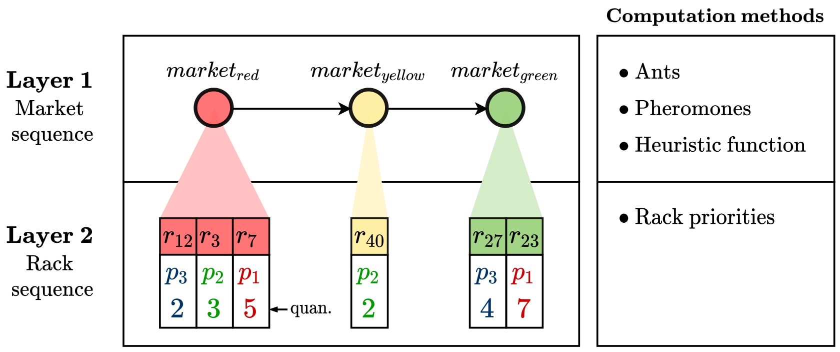


Figure 9.8: Illustration of the two layered pick route determined by one ant of the algorithm. A pick route consists of a market sequence and a rack sequence. The market sequence stores a list of visited markets while the rack sequence stores the rack id in combination with the picked amount of the product from this rack.

(2) Racks located closer to the sub aisle's entrance should be visited second.
(3) Racks that provide the largest quantities should be visited third.

9.4.4 Heuristic Function

To identify the most promising paths and assess the attractiveness of a market, the ants apply a heuristic function. We base the attractiveness of a market on two factors: (i) the closeness of the market to the ant's current location, and (ii) the availability of required items. Thus, we define the heuristic function as follows:

$$\eta^k_{m,n} = \left(\frac{1}{d_{m,n}}\right)\left(I^k_n\right) \text{ , where } n \in U^k \tag{9.6}$$

where $\eta^k_{m,n}$ is the heuristic value that the ant k currently located at market m associates with the edge (m, n) leading to market n. U^k is the set of markets the ant has not visited yet and $d_{m,n} > 0$ refers to the Manhattan distance between the markets. $I^k_n \in [0, 1]$ denotes the percentage to which the required items of ant k are available at market n. The higher the heuristic value, the more attractive is the market for the ant.

9.4.5 Objective Functions

After retrieving possible pick routes from the algorithm, we use two objective functions to asses the quality of the route.

9.4.5.1 Travel Distance

This objective function calculates the travel distance of a pick route and measures the extent to which the soft constraints SC_1 and SC_2 are satisfied. We define a pick route P to be $P = (M, R)$ where $M = (m_1, ..., m_k)$ refers to the market sequence and $R = (r_1, ..., r_l)$ refers to the rack sequence. We then define the objective function as follows:

$$travelDistance(P) = d^{pd}_{m_1} + \sum_{i=1}^{l} d^{sub}_{r_i} + \sum_{i=1}^{k} d^{cross}_{m_i} + \sum_{i=1}^{k-1} d^{market}_{(m_i,m_{i+1})} + d^{pd}_{m_k} \quad (9.7)$$

where we sum up the distance from the start p/d-point to the first market ($d^{pd}_{m_1}$), the sum of the distances within each entered sub aisle ($d^{sub}_{r_i}$), the sum of the distances within the cross lanes ($d^{cross}_{m_i}$), the distances between the visited markets ($d^{market}_{(m_i,m_{i+1})}$), and the distance from the last visited market to its closest p/d-point ($d^{pd}_{m_k}$).

9.4.5.2 Weight Violation

The second objective function measures the extent to which a pick route satisfies the soft constraint SC_3 and counts the number of weight violations in the product picking sequence. A weight violation occurs if a heavy product is collected after a much lighter product. In this case, the order picker must rearrange the lighter products already placed on the picking cart to prevent damage. The user-specified threshold `allowedWeightDifference` defines the acceptable weight difference between the heavier and the lighter products. Using this threshold, we count the number of weight violations in a product picking sequence.

9.4.6 ACO Algorithm Procedure

This section presents our proposed ACO algorithm and shows the pseudo-code in Algorithm 8. The algorithm receives the state of the warehouse as well as the pick list as input parameters. Further parameters are the maximum number of iterations without improvements (`maxIterWoImpr`), the maximum number of cataclysms (`maxCataclysms`), and the maximum number of iterations (`maxIter`). After the computation, the algorithm returns a list of optimized pick routes (`pickRoutes`). First of all, the algorithm constructs the graph (line 1) and initializes the pheromones (line 2). The algorithm initializes pheromones with their maximum possible value determined by τ_{max}. Additionally, a minimum pheromone can be specified by using the value τ_{min} in the parametrization of the algorithm. Then, a while loop starts and uses the concept of cataclysms [CWQX13] and a maximum number of iterations as stopping criterion (line 3): The parameter `maxCataclysms` specifies the maximum number of cataclysms that may occur. The parameter `maxconsIterWoImpr` defines the time window in which the ACO algorithm must improve the current Pareto front to prevent the cataclysm operator from being applied. The parameter `maxIter` defines the maximum allowed number of iterations regardless of happened cataclysms.

Algorithm 8: Proposed ACO Algorithm.

```
   Input: warehouseState, pickList
   Parameter: maxIterWoImpr, maxCataclysms, maxIter
   Output: pickRoutes
 1 graph = constructGraph()
 2 pheromones = initializePheromones()
 3 while cataclysms < maxCataclysms || iter < maxIter do
 4     iter++
 5     pickRoutes = constructPickRoutes()
 6     $pickRoutes_{ib}$ = selectParetoPickRoutes(pickRoutes)
 7     $pickRoutes_{merged}$ = $pickRoutes_{ib} \cup pickRoutes_{gb}$
 8     $nextPickRoutes_{gb}$ = selectParetoPickRoutes($pickRoutes_{merged}$)
 9     updatePheromones()
10     if isParetoFrontImproved() then
11         $pickRoutes_{gb}$ = $nextPickRoutes_{gb}$
12         consIterWoImpr = 0
13     else
14         consIterWoImpr++
15         if consIterWoImpr >= maxIterWoImpr then
16             $pickRoutes_{cataclysm}$ = $pickRoutes_{cataclysm} \cup pickRoutes_{gb}$
17             resetPheromonesOnGlobalBestRoutes()
18             cataclysms++
19             consIterWoImpr = 0
20 $pickRoutes_{cataclysm}$ = $pickRoutes_{cataclysm} \cup pickRoutes_{gb}$
21 return selectParetoOptimalPickRoutes($pickRoutes_{cataclysm}$)
```

Inside the loop, the algorithm increases the number of current iterations (line 4) and constructs pick routes (line 5). The general idea is to place one ant on each market of the graph from which the ant starts to create a pick route. The next market is selected based on the pheromone values and the heuristic function. We propose two different versions of the ACO to combine these values as explained later. For each found pick route, the algorithm computes the reverse pick route by reversing the market sequence, toggling the sides from which the ant entered the markets, and recalculating the rack sequence. We store the pick routes the ants construct in each iteration in the variable `pickRoutes`. In the next step, the algorithm selects the Pareto-optimal pick routes of this iteration by calculating the objective function and the Pareto rank of all routes. Afterward, the iteration-best ($\texttt{pickRoutes}_{\texttt{ib}}$ in line 6) and the global-best pick routes ($\texttt{nextPickRoutes}_{\texttt{gb}}$ in line 7) are merged into a single set and the Pareto-optimal pick routes in this set represent the next set of global-best pick routes. The algorithm then uses the iteration-best pick routes and the global-best pick routes to perform the pheromone update in line 9, which we explain later in Section 9.4.7 and Section 9.4.8. In the further course of the iteration, the ACO algorithm checks whether the cataclysm operator must be applied and compares the global best pick routes of the last and the current iteration (line 10). If the ACO algorithm succeeded in improving the Pareto front, the algorithm updates the set $\texttt{pickRoutes}_{\texttt{gb}}$ in line 11, and resets the counter variable `consIterWoImpr` to 0 in line 12. However, if no improvement was made (line 13), the algorithm increments this counter variable (line 14). If multiple consecutive iterations fail to achieve an improvement (line 15), the search is considered stuck, and the cataclysm operator is applied. In case the cataclysm is applied, the algorithm includes global-best pick routes $\texttt{pickRoutes}_{\texttt{gb}}$ into the set $\texttt{pickRoutes}_{\texttt{cataclysm}}$ (line 16), resets the pheromones on the edges representing the pick routes in $\texttt{pickRoutes}_{\texttt{gb}}$ to the lowest possible value (line 17), and empties the set $\texttt{pickRoutes}_{\texttt{gb}}$. Then, the algorithm increments the number of cataclysms in line 18 and resets the counter variable `consIterWoImpr` to 0 in line 19. After the main loop terminates, the algorithm includes the global-best pick routes $\texttt{pickRoutes}_{\texttt{gb}}$ of the last iteration into the set $\texttt{pickRoutes}_{\texttt{cataclysm}}$ in line 20 and returns the Pareto-optimal pick routes in this set (line 21).

9.4.7 ACO$_3$ Variant

Since the handling of multi-objective problem statements using ACO is non-trivial, we introduce two variants of our algorithm that handle pheromone updates distinctly. Both variants are inspired by [ASG07] who propose four different variants to handle multi-objective problems with an ACO. We select the two best performing variants (ACO$_3$ and ACO$_4$) and integrate them in our approach to compare which variant produces the best results in our problem domain. This section introduces the ACO$_3$ variant, which applies one ant colony using a single pheromone matrix τ for optimizing both objectives simultaneously. In each construction step, the probability of selecting an edge is defined as:

$$prob^k_{m,n} = \frac{(\tau_{m,n})^\alpha(\eta^k_{m,n})^\beta}{\sum\limits_{n \in U^k}(\tau_{m,n})^\alpha(\eta^k_{m,n})^\beta}, \text{ where } n \in U^k \tag{9.8}$$

where $prob^k_{m,n}$ denotes the probability of ant k located at market m to select the edge (m,n) leading to market n. $\tau_{m,n}$ refers to the pheromone value of edge (m,n). $\eta^k_{m,n}$ denotes the heuristic value (see Formula 9.6) that the ant associates with the edge (m,n). The parameters α and β control the importance of the pheromone and heuristic values. Lastly, U^k represents the set of markets that ant k has not visited yet.

When performing the pheromone update, the ACO$_3$ variant rewards the iteration-best pick routes in 90% of the time and the global-best pick routes (found since the last cataclysm) in 10% of the time to update the pheromone matrix τ. We update the values according to the following rule [ASG07]:

$$\tau_{m,n} = (1-\rho)\cdot\tau_{m,n} + \Delta\tau_{m,n} \tag{9.9}$$

$$\Delta\tau_{m,n} = \begin{cases} 1, & \text{if } (m,n) \text{ belongs to a pick route in } PF \\ 0, & \text{otherwise} \end{cases} \tag{9.10}$$

where ρ refers to the evaporation factor and $\Delta\tau_{m,n}$ is the amount of pheromone that will be added to the edge (m,n). PF refers to the Pareto front containing the solutions to be rewarded.

9.4.8 ACO$_4$ Variant

The ACO$_4$ variant also applies one ant colony but a pheromone matrix τ^1 for optimizing the first objective function, and another pheromone matrix τ^2 for optimizing the second objective function. When deciding which edge to explore next, an ant randomly selects a pheromone matrix using a uniform distribution. In each construction step, we calculate the probability of selecting an edge as [ASG07]:

$$p^k_{m,n} = \frac{(\tau^i_{m,n})^\alpha(\eta^k_{m,n})^\beta}{\sum\limits_{u \in U^k}(\tau^i_{m,n})^\alpha(\eta^k_{m,n})^\beta}, \text{ where } n \in U^k \text{ and } i \in \{1,2\} \tag{9.11}$$

where $\tau^i_{m,n}$ refers to the pheromone value of edge (m,n) with regards to the pheromone matrix τ^i. At the end of an iteration, the ACO$_4$ variant updates the pheromone matrix τ^i by rewarding the iteration-best pick route PR^i_{ib} that minimizes the objective function of_i [ASG07]:

$$\tau^i_{m,n} = (1-\rho) \cdot \tau^i_{m,n} + \Delta\tau^i_{m,n} \tag{9.12}$$

$$\Delta\tau^i_{m,n} = \begin{cases} \frac{1}{1+of_i(PR^i_{ib})-of_i(PR^i_{gb})}, & \text{if } (m,n) \text{ belongs to } PR^i_{ib} \\ 0, & \text{otherwise} \end{cases} \tag{9.13}$$

where ρ refers to the evaporation factor and $\Delta\tau^i_{m,n}$ represents the pheromones added to the edge (m,n) in pheromone matrix τ^i. PR^i_{gb} refers to the global-best pick route minimizing the ith objective function of all pick routes constructed since the last cataclysm occurred.

9.5 Summary

In this chapter, we proposed our main contributions focusing on **Goal B**: *Improving the quality of optimization strategies in complex systems-of-systems with a special attention to the field of logistics*. We answer **RQ B.2** and its subordinate questions **RQ B.2.1** and **RQ B.2.2** by proposing an integrated approach to optimize horizontal systems-of-systems. By applying the approach to the use case of storage assignment and order picking in mezzanine warehouses, we analyze the interrelation of the two parallel processes. In addition to executing the proposed approaches once, they can also be integrated into a feedback loop to adapt the solution to a changing environment. Therefore, we consider the approaches proposed in this chapter as part of the adaptation planning layer. The findings of this chapter can be used in the future to adapt the framework from Chapter 7 to cope with horizontal systems-of-systems and to be applicable to a wider range of use cases.

Part III

Evaluation

Chapter 10

Self-aware Optimization Framework

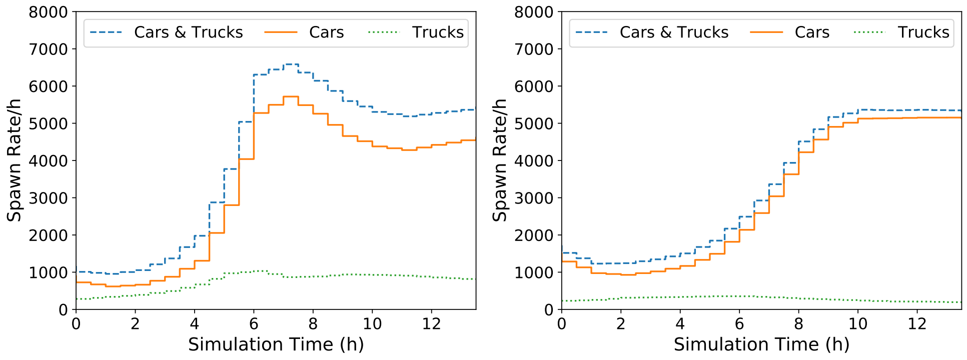
In this chapter, we evaluate our self-aware optimization framework introduced in Chapter 7. This chapter is based on our publications [LHKK21a, LHKK21c]. In line with the running example of this chapter, we also evaluate the framework on the prototypical use case platooning coordination. Still, the contribution of this chapter can be transferred to other domains where the application of SAS concepts appears meaningful. We simulate a real road section of the German A8 highway and use real traffic observed by the Federal Highway Research Institute to apply scenarios that are as realistic as possible. We apply the framework to two scenarios covering a weekday and a weekend day to analyze performance under different circumstances.

In this chapter, we first define the applied evaluation methodology with scenarios, simulation setup, and configurations in Section 10.1. Then, we evaluate the situation detection, strategy selection, and parameter optimization component in Section 10.2, Section 10.3, and Section 10.4, respectively. In Section 10.5, we assess the performance of the entire framework against the platooning metrics and analyze the performance over the course of simulation. Finally, Section 10.7 summarizes our results and refers to the research questions.

10.1 Methodology

In this work, we use the platooning coordination use case as a running example of our self-aware optimization framework. In this context, we also evaluate our framework in this use case. We first define the applied scenarios, then summarize the testbed and specify the framework configuration for our evaluation before proposing our baseline approaches.

We use a simulated road section of the German highway A8, which extends from the Stuttgart interchange to the Stuttgart-Degerloch exit. According to Süddeutsche Zeitung, this section is one of the busiest highway sections in Germany [dpa]. In addition to the realistic model of this highway section, we use real traffic data provided by the Federal Highway Research Institute of



(**a**) Wednesday (**b**) Saturday

Figure 10.1: Considered traffic scenarios of the framework evaluation for Wednesday on the left and Saturday on the right. Total number of spawning vehicles is depicted as blue dashed line, cars are depicted as solid orange line, and trucks are depicted as dotted green line.

Germany [bas] to define the vehicle spawn rates for our simulation. After a detailed analysis of the traffic values for each day of the week with the goal of selecting two distinct days with individual traffic volume profiles, we selected Wednesday as the representative weekday, and Saturday as the representative weekend day. Figure 10.1 shows the traffic volume for the selected days between 12:00 AM and 2:00 PM. As the simulation of such high traffic volume requires high computational power and a long computation time, we decided to simulate the first 14 hours of a day. This time interval contains a typical traffic volume profile for weekdays and weekends and, therefore, provides a good balance between long runtime and comprehensive simulation. We set the platooning percentage of all vehicles to 70% as we assume that not every vehicle is capable of platooning or drivers choose not to participate. Furthermore, we set the maximum speed limit of cars to 120km/h, which corresponds to the actual speed limits on this section [Bre]. In our evaluation, we use two types of situation detection—OPTICS and rule-based situation detection—and two types of triggers for strategy selection—HV- and threshold-based triggers—which results in four simulations per traffic profile. Since our approach involves Bayesian Optimization that incorporates randomness, we run three different random seeds in the traffic simulator SUMO for each simulation. We set the number of seeds, that is, the number of repetitions for each configuration, to three as the runtime for a single scenario is around 9 days.

We perform our simulations in the cloud of the Chair of Computer Science II at the University of Würzburg. This cloud consists of 18 hosts, each running RHEL-7-8.2003.0.el7.centos and oVirt Node 4.3.10 with Kernel-based Virtual Machines (KVM) version 2.12.0. The cloud contains one large ProLiant DL380 Gen9 host with two Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz CPU sockets and eight cores per socket. The remaining hosts are ProLiant DL160 Gen9 type with two CPU sockets of type Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz, eight cores per socket, and two CPU threads per core. We use three identical virtual machines for the simulations, which are deployed in our private cloud. Each virtual machine has two CPU sockets, each with 4 cores running at 2.6 GHz and 32 GB available RAM. We measure the simulation runtime of our scenarios, resulting in an average runtime of 9.5 days for the Wednesday scenarios and 9 days for Saturdays. Since the traffic volume on Saturdays is lower than on Wednesdays, these simulations require less runtime. Due to the long duration of a single scenario run of about nine days and the limited time for evaluation, we only performed three replicates per scenario, which unfortunately does not allow for statistical significance tests. However, additional runs may be conducted in the future to confirm the results presented.

We test two situation detection approaches and two triggers for strategy selection, which we now define. For the situation detection, we use a rule-based approach as well as OPTICS, which we have already presented in Section 7.5. As data input for the situation detection we use the amount of vehicles on the road. We derived the rules for the rule-based approach by taking definitions for peak hours, medium, and low traffic volumes from the German city of Rostock [AG]. This study states that peak hours occur from Monday to Friday, which led us to the decision to use the highest traffic volume on Saturday as the upper limit for off-peak hours. This results in the following rules: We consider the first situation with lowest traffic volume, where the maximum number of vehicles on the road section is 120. We define the medium traffic volume from 121 to 280 vehicles and define the peak traffic volume above 280 vehicles on the road segment. OPTICS requires the definition of the minimum number of points and the minimum cluster size, both of which we set to a value of 45. We determined this value in a preliminary study with different parameters, which showed that this configuration is best suited for our use case.

Similar to the situation detection, we also evaluate two triggers for the strategy selection component: HV and individual thresholds. Both methods incorporate the four objective metrics introduced in Section 6.1.1 to assess the performance of the currently active strategy: (i) throughput, (ii) time loss, (iii) platoon utilization, and (iv) platoon time. The HV requires the definition of a reference value which we set to -0.1, which is outside the range of values of all considered platoon metrics. We set the HV threshold to 0.3 and consider a time window size of five, in which the HV must fall below the threshold at least three times to trigger the strategy selection. The threshold-based trigger requires the definition of an individual threshold per platoon metric, which we set as follows. We set the throughput metric threshold to 0.5, since we assume that platooning coordination strategies have little impact on this metric. Furthermore, we set the threshold for the time loss metric to 0.9, since our preliminary study showed that the time loss metric was always above 0.85 for all runs, so we need a very strict threshold to have any effect at all. We set the platoon utilization metric to 0.62, which is also close to the defined platoon percentage and should lead the system to high platoon utilization. The threshold for the platoon time is 0.3, a comparatively low value that provides the framework with a large margin for testing different strategies. The strategy selection component requires specifying the number of optimization cycles for each strategy as initial trial phase in which no other strategy can be selected. We set this value to ten. Finally, we specify the order in which the platooning coordination strategies are selected: Best Distance (BD), Best Velocity (BV), as well as Best Distance and Lane (BDL). In our study regarding the situation-awareness of platooning coordination strategies [LNH$^+$21] we analyzed that the BV strategy is the most appropriate for this use case. This would mean that this strategy should to be tested first. However, we decided to start with the BD strategy to force the framework to select another strategy.

To evaluate the performance of our framework against a set of baseline approaches, we apply the BD, BV, and a rule-based strategy to the two scenarios. Table 10.2 summarizes the configurations of our baseline strategies. We derived the rule-based strategy from our study of the self-awareness of strategies [LNH$^+$21]. Since the BV strategy performs by far the best in this study, we distinguish two cases in which we change the configuration depending on the number of vehicles on the road and the average car speed. The rule-based strategy uses the first configuration when the number of vehicles is below 500 and the average car speed is above 125 km/h. It applies the second configuration when the number of vehicles is higher than 500 and the average car speed is lower than 125 km/h. We also apply the same set of rules as fallback-mechanism in our framework when the applied situation detection cannot detect the current situation.

Table 10.1: Configuration of the framework and tested strategies, algorithms, and methods used in the evaluation. The HV trigger of the strategy selection component require definitions of further parameters such as reference values, thresholds, time windows, and threshold exceeds. The individual threshold trigger requires the definition of thresholds for all used metrics.

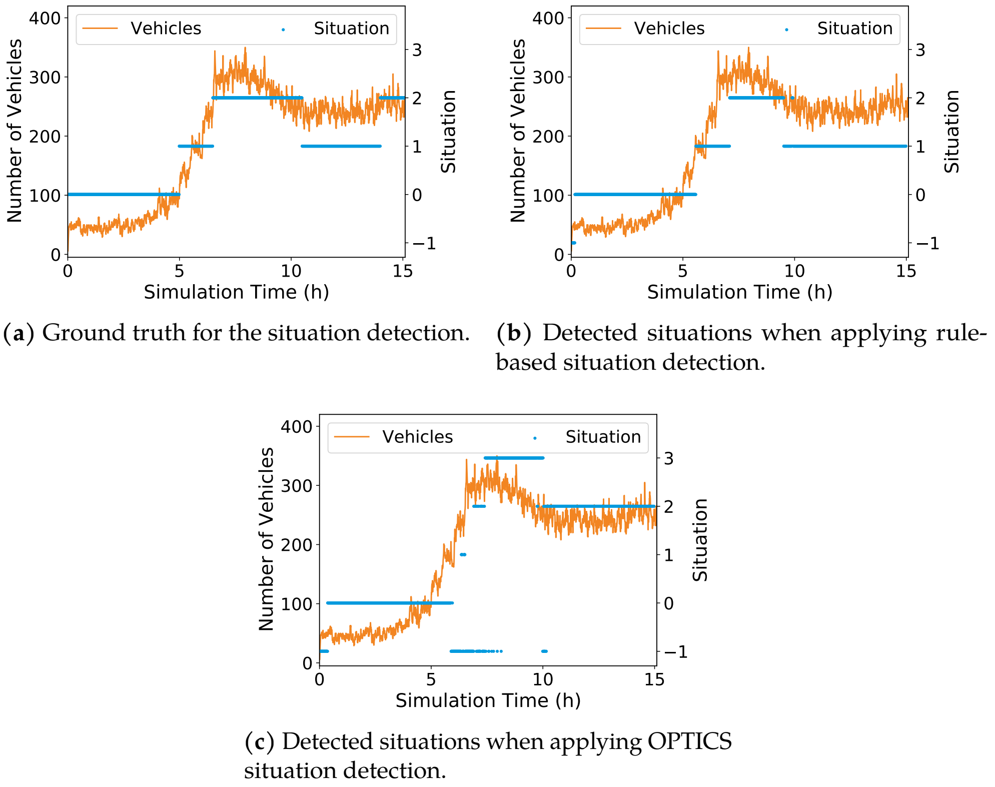
DDM Part	Parameter	Value
Use Case	Available strategies	BD, BV, BDL
Situation Detection	Algorithm	RuleBased, OPTICS
Strategy Selection	Method	HV, threshold
	Min. opt. attempts	10
HV	Reference values	-0.10
	Threshold	0.30
	Time window size	5
	Threshold exceeds	3
Thresholds	Throughput	0.50
	Time loss	0.90
	Platoon utilization	0.62
	Platoon time	0.30

Table 10.2: Configurations of the baseline approaches used in the evaluation. All strategies require the definition of advertising duration and lane speed thresholds. The BD strategy requires the maximum speed difference, BV and both rule-based configurations require the search distance front and back.

Parameter Name	BD	BV	Rules I	Rules II
Advertising duration [m]	10	10	10	5
Search distance front [m]	-	600	600	400
Search distance back [m]	-	250	250	200
Max. speed difference [km/h]	35	-	-	-
Speed threshold lane 2 [km/h]	100	100	100	100
Speed threshold lane 3 [km/h]	130	130	130	130
Speed threshold lane 4 [km/h]	160	160	160	160

10.2 Evaluation of the Situation Detection Component

In line with the workflow of our optimization framework, we start our evaluation with the situation detection component. Keep in mind, that this component uses the current amount of vehicles on the road to identify a situation. Therefore, we analyze the detected situations during the simulation for both scenarios and compare the rule-based and OPTICS approaches to the ground truth. Figure 10.2 shows the ground truth for situation detection and the results of the component applied to the Wednesday scenario. The orange line represents the vehicle spawn rate, while the blue dots represent the cluster ID, that is, the detected situation, at a given time. A feature of clustering algorithms such as



(**a**) Ground truth for the situation detection.

(**b**) Detected situations when applying rule-based situation detection.

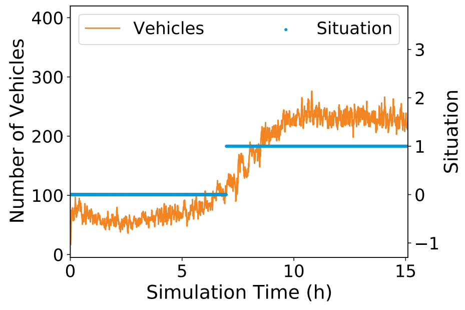
(**c**) Detected situations when applying OPTICS situation detection.

Figure 10.2: Actual situations of the ground truth and detected situations of the rule-based and OPTICS approach for Wednesday traffic data. The orange line represents the vehicle spawn rate at a specific point in time. The blue dots represent the detected situation at the current point in time incorporating all previously observed data points.

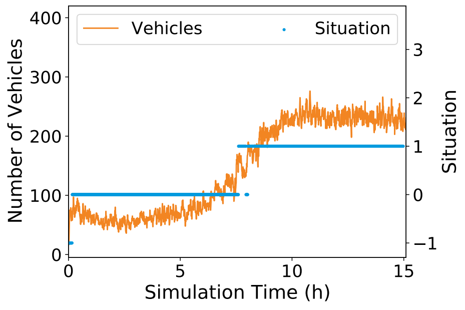
OPTICS is that the identified clusters and observations assigned to them might change as new measurements considered. This can cause the cluster IDs for an observation to change over time, which is the motivation of our ongoing model learning approach in the situation detection component (c.f. Section 7.5). However, this behavior is not part of the illustration in Figure 10.2. The figure shows the cluster numbers assigned when the observation first occurred. This represents the situation based on which the framework makes its decisions. The figures 10.2a and 10.2b show that the rule-based situation detection component is close to ground truth, as it identifies all three situations, but assigns fewer observations to the peak traffic cluster. In addition, the rule-based approach does not detect the start of the second peak traffic cluster. The good performance of this approach was expected since the rules were derived from the ground truth. The situation detection using OPTICS, as shown in Figure 10.2c, identifies the situations using clustering mechanisms and identifies four different situations, but is not able to cluster all observations denoted by cluster ID -1. The four identified situations are less evenly distributed in terms of the observations they contain, as cluster number one contains only a few observations. Nevertheless, this mechanism is able to distinguish different situations even if they do not completely consistent with the ground truth.

The results of the situation detection component applied to the Saturday scenario are depicted in Figure 10.3. Again, the orange line represents the vehicle spawn rate and the blue dots represent the identified cluster ID. Similarly to the Wednesday scenario, the rule-based approach is close to the ground truth, which is not surprising since the rules were derived from the ground truth. However, the OPTICS approach shows a different behavior as it is not able to identify at least two different situations and combines all observations into one situation. The poor performance of this approach could be due to an unfavorable parameter configuration resulting from our preliminary parameter study. Another factor could be the lower number of vehicles on the road compared to the Wednesday scenario, which could lead to similar observation data. Further evaluation using more extensive scenarios and additional parameter studies may provide more insight in the future.

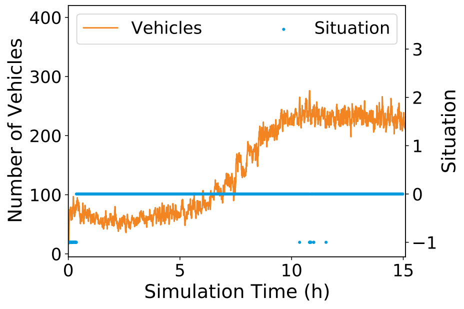
In summary, this evaluation shows that the rule-based approach performs well against the defined ground truth for both scenarios. The OPTICS approach identifies distinct situations in the Wednesday scenario, but only a single situation for the Saturday scenario. The ground truth derived rules work predictably well, but are a very rigid approach and do not provide flexibility for future changes. A rule set must be defined at design time using expert knowledge and will not be further adapted. On the other hand, the clustering approach



(**a**) Ground truth for the situation detection.



(**b**) Detected situations when applying rule-based situation detection.



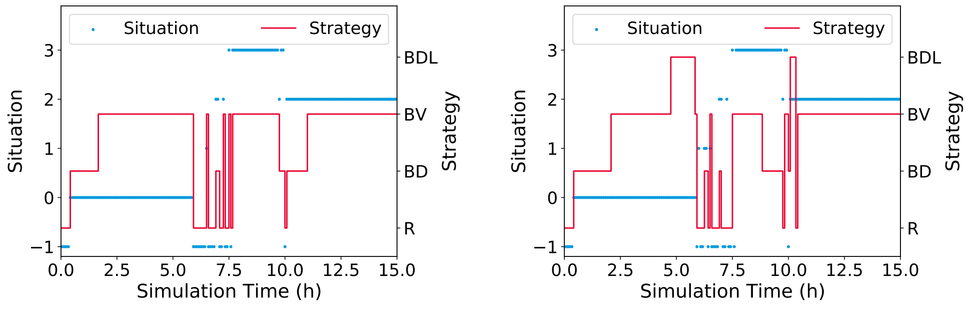
(**c**) Detected situations when applying OPTICS situation detection.

Figure 10.3: Actual situations of the ground truth and detected situations of the rule-based and OPTICS approach for Saturday traffic data. The orange line represents the vehicle spawn rate at a specific point in time. The blue dots represent the detected situation at the current point in time incorporating all previously observed data points.

OPTICS provides more flexibility, but does not find the situations defined in the ground truth as reliably. For the future, extended simulations with, for example, several days could reveal more potential for improvements. In addition, rule learning methods could be used to adapt the rule-based situation detection during runtime.

10.3 Evaluation of the Strategy Selection Component

The second component of the framework, the strategy selection component, receives the detected situation and selects the most promising strategy to be applied in the adaptation planning system. In this section, we analyze the proper



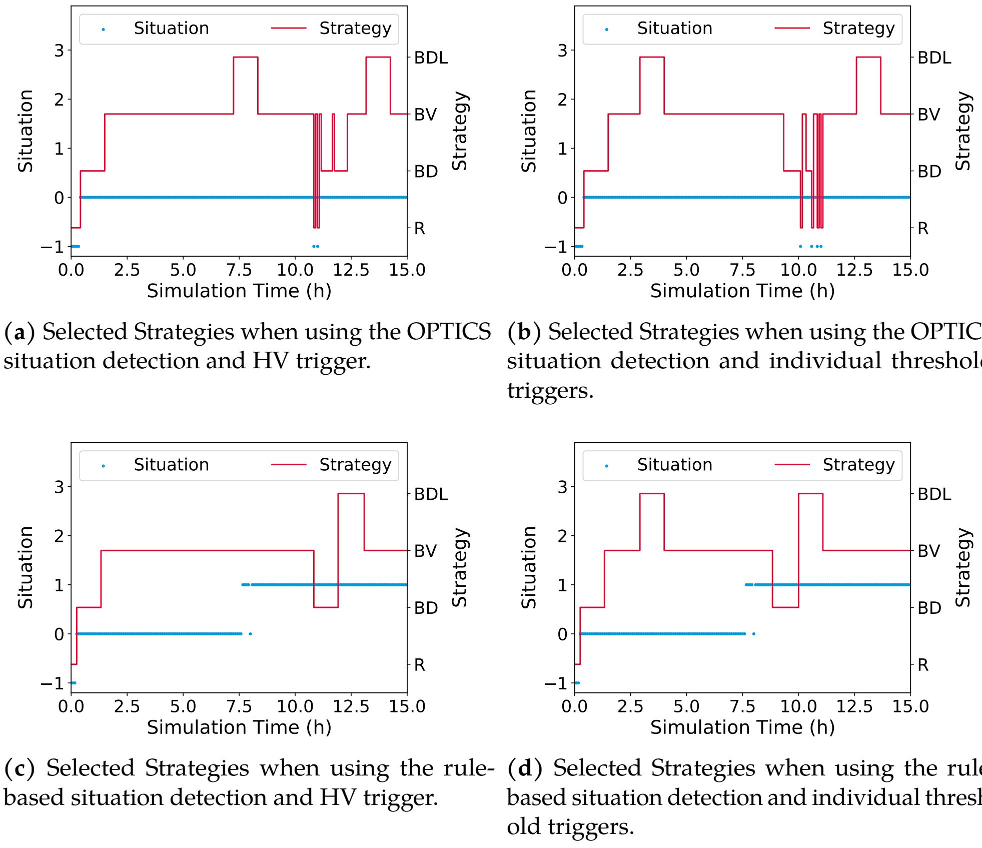
(**a**) Selected Strategies when using the OPTICS situation detection and HV trigger.

(**b**) Selected Strategies when using the OPTICS situation detection and individual threshold triggers.

Figure 10.4: Strategy selection on Wednesday traffic data. Blue points represent the detected situation at a specific point in time. The red line represents the selected adaptation planning strategy at a specific point in time. (R = *Rules*, BD = *BestDistance*, BV = *BestVelocity*, and BDL = *BestDistanceAndLane*)

operation and performance of the strategy selection component. Therefore, Figure 10.4 shows the selected strategies for the Wednesday scenario using OPTICS as the situation detection mechanism and the HV trigger in Figure 10.4a as well as the individual thresholds as trigger in Figure 10.4b. We decided to use continuous line charts with vertical lines representing a strategy change to better visualize the changed strategies especially in cases where the selection changes back and forth frequently. We base this evaluation solely on OPTICS, as it identifies different situations for the Wednesday scenario and is able to handle new situations not defined in a rule set.

The blue points represent the determined situation, while the red line illustrates the selected strategy at a certain point in time, that is, the height of the line represents the selected strategy. The left figure shows that the strategy selection component selects a strategy and switches to the next one if the performance metrics fall below the thresholds and the triggers activate the selection. When using the HV trigger, the strategy selection remains at the BV and does not switch to the BDL within the first six simulation hours compared to the individual threshold trigger. After this time, the observations are classified as noise by the situation detection, which causes the strategy selection to revert to the rule-based strategy. Whenever new situations occur, the strategy selection starts with the BD and tests its performance before switching to the BV strategy. The results show that the individual thresholds trigger the strategy selection more often than to the HV trigger because the selection component examines



(**a**) Selected Strategies when using the OPTICS situation detection and HV trigger.

(**b**) Selected Strategies when using the OPTICS situation detection and individual threshold triggers.

(**c**) Selected Strategies when using the rule-based situation detection and HV trigger.

(**d**) Selected Strategies when using the rule-based situation detection and individual threshold triggers.

Figure 10.5: Strategy selection on Saturday traffic data. Blue points represent the detected situation at a specific point in time. The red line represents the selected adaptation planning strategy at a specific point in time. (R = *Rules*, BD = *BestDistance*, BV = *BestVelocity*, and BDL = *BestDistanceAndLane*)

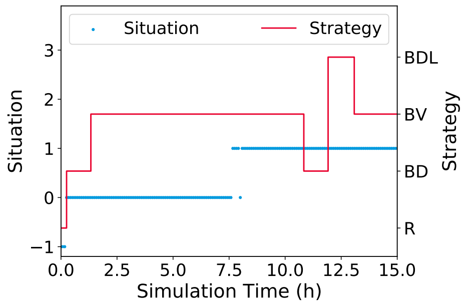
the BDL twice. This is the intended behavior of the framework and tells us that it is working properly. Also, this may indicate that the individual thresholds are too restrictive and could be relaxed to avoid jitters between strategies.

Figure 10.5 shows the results of the strategy selection component for the Saturday scenario using OPTICS and rule-based situation detection in combination with the HV and individual threshold triggers. The reason for using the rule-based situation detection in this evaluation is that OPTICS situation detection was not able to identify more than one situation for the Saturday scenario. Figure 10.5a presents the OPTICS and HV evaluation, Figure 10.5b presents the OPTICS and individual threshold evaluation, Figure 10.5c illustrates the rule-based and HV evaluation, and Figure 10.5d shows the rule-based and individual threshold evaluation. Again, the blue points represent the identified situation, and the red line represents the selected strategy at a given time. All figures show the desired exploratory behavior of the strategy selection when a new situation occurs due to the step-wise strategy change at the beginning. If a strategy performs well, it is not replaced and remains active until the triggers indicate a performance degradation. Since the OPTICS situation detection identifies only one situation and classifies some observations as noise, it shows a clear step-wise strategy change and a reversion to the rule-based strategy when the situation detection reveals noise. When using the rule-based situation detection, the strategy selection is more stable since no fallback mechanisms are required. However, Figure 10.5c shows an anomaly in the strategy selection behavior, as the detection of a new situation does not trigger a new exploration of strategies after around eight hours. A detailed analysis of this behavior led us to the conclusion that the detection of a situation change was not perfectly aligned with the strategy selection component and, hence, resulted in a lost situation change. Thus, the currently active strategy, that is, the BV, remains active until about eleven hours of simulation time. At this point, the HV trigger indicates a performance degradation of the current strategy and the strategy selection selects the BD strategy. However, it is discarded after the initial trial period and the strategy selection switches to the BDL strategy. The same lost update of a new situation can be observed in Figure 10.5d. However, this figure shows a faster discarding of the currently active strategy, similar to the behavior in Figure 10.5b. This also indicates that the individual thresholds might be too restrictive and could be relaxed in the future to produce a more stable result.

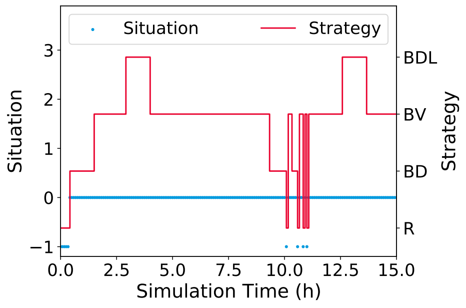
In summary, this evaluation shows that both algorithm selection trigger methods work properly and activate the algorithm selection when the performance of the currently active strategy deteriorates. While the HV threshold provides a more stable result, the individual thresholds appear to detect performance degradation earlier. Therefore, the individual thresholds explore more possible strategies, but also result in higher jitter compared to the HV. However, the definition of the individual thresholds can be adjusted in future evaluation studies to achieve a trade-off between detecting performance degradation quickly and reducing jitter. All in all, both methods work properly and are capable of triggering the algorithm selection.

10.4 Evaluation of the Parameter Optimization Component

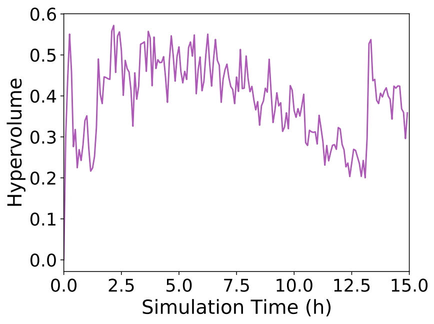
We evaluate the performance of our optimization component by analyzing the course of the HV metric used by this component to optimize the parameter configuration of the current adaptation planning strategy. The HV metric (c.f. Section 3.3) accumulates the platooning metrics into one objective metric that can be used by the single-objective Bayesian Optimization. Figure 10.6 shows evaluations of the Saturday scenario using rule-based situation detection and HV as trigger for the strategy selection component on the left (Figure 10.6a and Figure 10.6c). The right side of the figure shows measurements for the Saturday scenario using OPTICS as situation detection mechanism and individual thresholds as triggers for strategy selection (Figure 10.6b and Figure 10.6d). The top figures show the identified situations in blue in combination with the selected strategies in red. The lower figures summarize the course of the HV metric, that is, the performance indicator of the platooning coordination strategy. The course of the HV metric appears to be very fluctuating for both configurations during the simulation time. This was to be expected, since the optimization component needs some time to learn which parameter setting works well for which strategy and situation. Therefore, it makes most sense to analyze time windows of the HV progression where the identified situation and strategy remain stable. This is also a reason for choosing Saturday scenarios for this evaluation, as traffic volumes do not fluctuate as much as in Wednesday scenarios, which allows for longer time frames per situation and strategy. When analyzing the first stable phase on the left between 2.5 and 7.5 hours of simulation time, the HV starts with a value of about 0.5 HV points and drops to 0.3 HV points. Then, it stabilizes back to about 0.5 HV points, indicating that the optimization component has explored different parameter settings and stabilized to a well performing set of parameters. As discussed earlier, the change in the situation is lost at about 7.5 hours of simulation time, resulting in a sharply decreasing trend in the HV. This leads to the extended HV threshold that triggers the strategy selection at about 11 hours of simulation time. The other configuration, depicted on the right of the figure, captures OPTICS and individual thresholds. In this evaluation, we can analyze the HV score for the simulation period starting at four hours up to eight hours of simulation time. The HV score shown on the bottom right starts at a low value of around 0.2 score points, but quickly increases to a value of 0.4 score points. This low start value is due to the recent strategy change from the BDL strategy which was discarded in favor of the BV strategy after its initial trial phase. After that, the HV score shows a slight increase to a value of about 0.58 score points, but then decreases again to values between 0.4 and 0.5 score points. This indicates, that



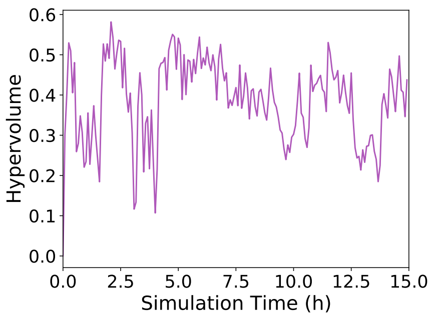
(**a**) Selected Strategies when using the rule-based situation detection and HV trigger.



(**b**) Selected Strategies when using the OPTICS situation detection and individual threshold triggers.



(**c**) HV score of the selected strategy when using the rule-based situation detection and HV trigger.



(**d**) HV score of the selected strategy when using the OPTICS situation detection and individual threshold triggers.

Figure 10.6: Evaluation of the optimization component on the Saturday scenario. The left side represents configurations using rule-based situation detection and HV triggers. The right side illustrates OPTICS situation detection and individual threshold triggers (R = *Rules*, BD = *BestDistance*, BV = *BestVelocity*, and BDL = *BestDistanceAndLane*).

the Optimization component finds better parameter settings for the selected strategy and then explores new parameter settings that unfortunately lead to worse HV values. This triggers the strategy selection, and since all existing strategies have already been explored, the fallback rules take place.

In summary, this evaluation shows us that the Optimization component has the potential to optimize the parameter settings of the adaptation planning strategies, as the HV score remains stable and shows slight increases in stable situations for situation and selected strategy. However, negative effects also occur when the Optimization component explores new parameter settings, which may lead to worse results compared to the previous settings that performed well. This indicates that the stable phases of identified situations and selected strategies, that is, the time for the Optimization component to optimize the parameter settings, may be too short to find stable configurations with good performance. Extended evaluations over several days or even weeks could provide more insight into the required amount of experience for the Optimization component and increase the overall performance of this component.

10.5 Evaluation of the Entire Framework

In our final evaluation, we analyze the overall performance of the framework. First, we compare the four defined configurations of the framework with the three baselines in terms of the four platooning metrics of throughput, time loss, platoon utilization, and platoon time. Table 10.3 presents the mean and standard deviation results for these metrics for both scenarios. We highlight the best values of each platooning metric for the baseline group and the framework group in bold. In both evaluation scenarios, the throughput metric results for all baselines and framework configurations are very close, with values between 0.9943 and 0.9952 and low standard deviations. In the Wednesday scenario, the BD baseline and rule-based situation detection combined with HV thresholds perform best on the throughput metric with values of 0.9952 and 0.9946, respectively. In the Saturday scenario, all configurations of the framework perform equally well, while the BV baseline performs best on the throughput metric with values of 0.9950 and 0.9951, respectively. All applied configurations and baselines show higher diversity for the time loss metric, ranging from 0.8992 to 0.9122 for Wednesday and from 0.9255 to 0.9411 for Saturday. Rule-based situation detection combined with individual thresholds performs best for this metric among all configurations tested, with a value of 0.9122 and 0.9333, but achieves a lower value compared to the BV baseline, with a value of 0.9199 and 0.9411 for Wednesday and Saturday, respectively. Results for the platoon utilization metric range from 0.6251 to 0.7176 and from 0.5999 to 0.7101 for Wednesday and Saturday, respectively. For this metric, the fallback rule baseline among the baselines and the OPTICS situation detection in combination with HV and individual thresholds perform best. Finally, the results for the platoon time metric range from 0.4908 to 0.6518 and from 0.4182 to 0.6199 for Wednesday and Saturday, respectively. Again, the fallback rules baseline performs best for both scenarios, and the OPTICS situation detection with HV and individual thresholds performs best among the framework configurations.

Table 10.3: Evaluation summary of the average and standard deviation for performance metrics throughput, time loss, platoon utilization, and platoon time for both scenarios. We compare four different configurations of the framework to three baseline mechanisms. The baseline mechanisms perform deterministic. The best values are shown in bold. (Hv = Hypervolume, Th = Threshold)

Scenario	Configuration	Throughput		Time Loss		Platoon Utilization		Platoon Time	
		mean	std	mean	std	mean	std	mean	std
Wednesday	Best Distance	**0.9952**	0.0	0.8992	0.0	0.6251	0.0	0.4908	0.0
	Best Velocity	0.9942	0.0	**0.9199**	0.0	0.6973	0.0	0.6109	0.0
	Fallback Rules	0.9950	0.0	0.9198	0.0	**0.7176**	0.0	**0.6518**	0.0
	OPTICS & Hv	0.9943	0.0003	**0.9122**	0.0022	**0.6690**	0.0030	**0.5442**	0.0090
	Rule-based & Hv	**0.9946**	0.0004	0.9102	0.0011	0.6647	0.0039	0.5302	0.0076
	OPTICS & Th	0.9945	0.0003	0.9110	0.0014	0.6566	0.0072	0.5275	0.0119
	Rule-based & Th	0.9943	0.0003	0.9108	0.0003	0.6343	0.0109	0.5005	0.0083
Saturday	Best Distance	0.9945	0.0	0.9255	0.0	0.5999	0.0	0.4522	0.0
	Best Velocity	**0.9951**	0.0	**0.9411**	0.0	0.6942	0.0	0.5833	0.0
	Fallback Rules	0.9950	0.0	0.9401	0.0	**0.7101**	0.0	**0.6199**	0.0
	OPTICS & Hv	0.9949	0.0001	0.9309	0.0004	0.6360	0.0019	0.4918	0.0022
	Rule-based & Hv	**0.9950**	0.0001	0.9297	0.0013	0.6367	0.0087	0.4880	0.0137
	OPTICS & Th	**0.9950**	0.0000	0.9323	0.0012	**0.6511**	0.0065	**0.5169**	0.0159
	Rule-based & Th	**0.9950**	0.0001	**0.9333**	0.0024	0.5677	0.0504	0.4182	0.0520

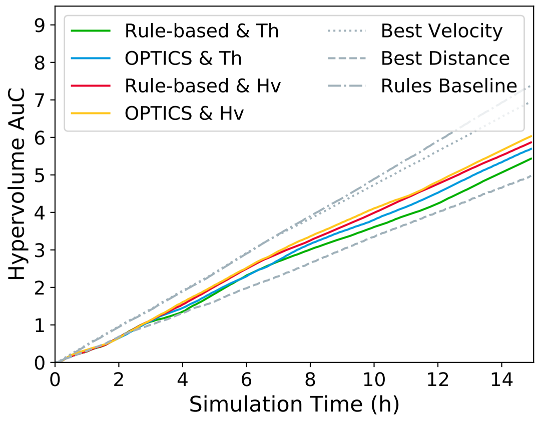


Figure 10.7: Mean area under curve evaluation over time for the HV score of all tested configurations and the baselines on the Wednesday scenario. The different colors represent the tested configurations, the x-axis shows the simulation time, and the area under curve is depicted on the y-axis.

The combination of the close average values for all metrics and the small standard deviations does not suggest significant advantages for some configurations. However, this indicates that the framework performs comparably well when considering the results of the baseline, which was designed and configured with complete prior knowledge based on the preliminary situation-dependency study we published [LNH+21].

In the following, we analyze the progression of the performance over the simulation time. Therefore, Figure 10.7 and Figure 10.8 present the mean HV area under curve over simulation time for all configurations and baseline strategies for Wednesday and Saturday. The baseline strategies are depicted as gray lines with a dotted line for the BV, a dashed line for BD and a dashed and dotted line for the rules baseline, while colors represent the configurations. Both plots show a similar result: The BV and rules baseline perform best, with a stable increasing gradient of the area under curve, while the BD baseline performs worst. The curves of the framework configurations do not increase at a constant rate, but show more fluctuations in the gradient. All lines are close to each other, but more differences appear as the simulation progresses. The OPTICS and rule-base situation detection with the HV trigger perform best for Wednesday. For the Saturday scenario, both configurations perform well, but OPTICS with individual thresholds outperforms them slightly beginning at ten hours simulation time. For both scenarios, the rule-based situation detection with individual thresholds performs worst.

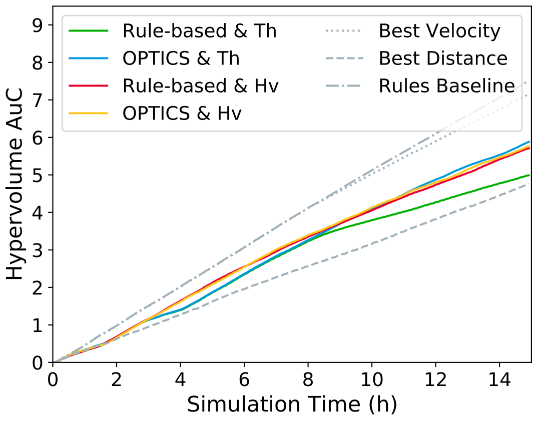


Figure 10.8: Mean area under curve evaluation over time for the HV score of all tested configurations and the baselines on the Saturday scenario. The different colors represent the tested configurations, the x-axis shows the simulation time, and the area under curve is depicted on the y-axis.

It is not surprising that the BV and the rules baseline perform best, since our use case study published in [LNH+21] extensively examined existing baseline strategies, their configuration, and their performance in various situations. Using this information, we then defined these baseline strategies to represent the best possible performance when complete knowledge of situations, strategies, and configuration was available at design time. However, such intensive studies are not feasible, especially in such dynamic, adaptive use cases. Moreover, it is in the nature of the framework to perform worse than the gold standard, since it needs some time to explore possible strategies and configurations before it can learn and profit from earlier decisions. The better performance of all framework configurations compared to the BD baseline shows that the framework is able to identify and select a strategy that works well. This reduces the need of expert knowledge or extensive case studies for a use case and, hence, provides a valuable contribution to self-aware optimization.

10.6 Threats to Validity

We identified the following threats to validity of our evaluation. First, the proposed framework and its components are evaluated on one road segment of the German highway which limits the transferability of our results. However, we used a real-world road segment with real-world traffic volumes to simulate challenges for the framework when applying it in reality as good as possible. Second, we simulated the traffic of the first fourteen hours of a day on Wednesday and Saturday. As discussed earlier we made this selection to limit the simulation time to around nine days but still keep the typical traffic profile for weekdays and weekends. Third, the small amount of repetitions per scenario due to the long simulation times do not allow for statistical tests to show the significance of our results. However, we presented mean and standard deviations of all scenarios and examine the performance from various points of view and compared it against a gold standard as well as typical strategies from the use case. We are convinced that this evaluation provides a first insight into the feasibility of the framework and its components and additional evaluation runs can be conducted in the future to validate the results. Finally, we decided to evaluate the framework in the platooning coordination use case as it provides a real-world example of SAS operating in a highly dynamic environment from the domain ITS. We are convinced that this use case is a meaningful selection to show the feasibility of the proposed framework. Still, the generalizability and transferability of the framework to other use cases in the ITS domain and also beyond needs to be assessed by applying it to other real-world use cases from other domains in future work.

10.7 Summary

In this chapter, we evaluated our main contribution, which focused on **Goal A**: *Self-aware optimization of adaptation planning strategies with particular attention to the field of ITS and logistics*. We answered **RQ A.6** and analyzed the component-based self-aware optimization framework for adaptation planning strategies on the use case of platooning coordination. First, we present the methodology of the evaluation, followed by an in-depth evaluation of the components of the framework. Then, we analyze the performance of the entire framework and show that it is able to analyze observations to identify the current situation and automatically select the most promising adaptation planning strategy. This reduces the need for expert knowledge and extensive case studies to configure an adaptation planning system for a specific use case.

Chapter 11

Vertical Systems-of-Systems Workflow

In this chapter, we evaluate our approach for tackling the rVRP from Chapter 8. This chapter is based on our accepted paper in the Springer Applied Intelligence Journal [LKK$^+$21a] and on our technical report [LKK$^+$21b]. Since our approach focuses on rVRP including a diverse set of constraints and requirements, we evaluate our approach on a set of real-world data. We retrieved this data from our cooperation company that provides intelligent logistic solutions for their customers, for example to plan routes for logistic service providers. In our evaluation we analyze the mean solution quality after the execution and the course of solution quality of our approaches and compare them to a set of state-of-the-art optimization techniques. We use the defined scores from Section 8.2.2 to assess the quality of computed solutions.

To this end, we first define the used data set as well as the defined problem instances on which we apply our approaches in Section 11.1. Afterward, we introduce the set of state-of-the-art optimization techniques we use for comparing our approaches in Section 11.2. Then, we discuss the evaluation procedure as well as the algorithm parametrizations in Section 11.3. We present our evaluation results separated by the problem instance in Section 11.4 and discuss threats to validity in Section 11.5. Finally, we summarize our findings and refer to the research questions posed in Section 11.6.

11.1 Problem Instances

Since we handle a real-world rVRP, we decided to use a real database for our evaluation instead of a benchmark instance as we require a huge level of detail for each order, vehicle, and driver. This would force us to adjust the available benchmark instances which would reduce the comparability of the results what is the main advantage of these instances. Therefore, our cooperation partner provided a database of real VRPs containing 30 vehicles with costs, capacities, and capabilities, 15 matching trailers with their specifications, and 30 drivers that can be assigned to vehicles with their individual capabilities. Further, the database contains three depots and 450 orders with locations around the German city Stuttgart. Unfortunately, we are not allowed to make this dataset publicly available since it is part of a non-disclosure agreement.

From this set of data, we define eight different problem instances for evaluating our proposed algorithms. In line with our separated handling of TSP and VRP instances—which refers to our solution for the vertical systems-of-systems structure of the problem—we decided to first evaluate the TSP-stage isolated and, afterwards, apply the algorithms on the VRP-stage that includes solving nested TSP instances. Instances without P&D behavior assume the pickup at the depot and only include the delivery planning at customers. For the evaluation of the TSP-stage, we define three problem instances:

1. A small problem instance of ten orders without P&D and without pause times (TSP-I),
2. a large problem instance of 30 orders without P&D and without pause times (TSP-II),
3. and a large problem instance of 30 orders without P&D but with pause times (TSP-II-P).

Similarly, we define three problem instances for evaluating the VRP-stage:

1. A small problem instance of 53 orders and five vehicles without P&D and pause times (VRP-I),
2. a small problem instances combined with pause times (VRP-I-P),
3. and a large problem instance of 100 orders, 13 vehicles without P&D and pause times (VRP-II).

Since we did not include P&D behavior, that is, each order has differing pickup and delivery stops, in the previous problem instances, we add two further instances that require P&D behavior:

1. A TSP problem instance with ten orders, one vehicle with P&D but without pause times (TSP-PD)
2. and a VRP problem instance with 62 orders, seven vehicles with P&D and pause times (VRP-PD).

We summarize all problem instances in Table 11.1. In all problem instances, the algorithms need to handle time windows for all orders. Using the real-world data explains the unusual amount of orders and vehicles since the minimum required vehicles depend on the characteristics of the orders. However, we only integrate pause times if we explicitly stated it, that is, in the problem instances TSP-II-P, VRP-I-P, and VRP-PD. The extension *P* of the problem instance label indicates that for this problem instance we add the following pause times: 9:30-10:00 AM, 11:30 AM-12:00 PM, and 2:30-3:00 PM. We consider static pause times to evaluate the ability of our algorithms to fulfill this requirement. However, also flexible pause times can easily be included to replace the static ones.

Table 11.1: Overview of the evaluated problem instances. We define three problem instances for the TSP and three for the VRP without P&D behavior. Finally, we define two problem instances covering P&D behavior.

Problem Instance	Orders	Vehicles	P&D	Pause Times
TSP-I	10	1	✗	✗
TSP-II	30	1	✗	✗
TSP-II-P	30	1	✗	✓
VRP-I	53	5	✗	✗
VRP-I-P	53	5	✗	✓
VRP-II	100	13	✗	✗
TSP-PD	10	1	✓	✗
VRP-PD-P	62	7	✓	✓

11.2 Alternative Algorithms for Comparison

We compare the performance of our algorithms (GA, ACO) against four alternative algorithms (Brute Force, Blackbox-I, Blackbox-II, and LS) from which the two Blackbox algorithms are provided from our cooperation partner. Since their provided algorithms are already implemented in OptaPlanner, we decided to also use OptaPlanner for an easy comparison of our new implementations. Hence, we implement our algorithms in the OptaPlanner Framework (cf. `https://www.optaplanner.org/`) using version *7.31.0.Final*. Of course it is possible to implement our approach without loss of functionality in other frameworks or, alternatively, completely independently.

Table 11.2 provides essential information on the functional requirements supported by each compared algorithm. First, we apply a deterministic Brute Force algorithm provided by OptaPlanner that supports all requirements of our

Table 11.2: Overview on the applied algorithms and their capabilities with respect to the requirements of the rVRP. We compare our algorithms to a standard Brute Force algorithm from OptaPlanner, two Blackbox algorithms provided by our cooperation partner, and a standard Tabu Search representative for Local Search.

Capabilities	Brute Force	Blackbox-I	Blackbox-II	LS	GA	ACO
Capacities	✓	✓	✓	✓	✓	✓
Setup Times	✓	✗	✗	✓	✓	✓
Time Windows	✓	✓	✓	✓	✓	✓
Tour Start Time Window	✓	✓	✓	✓	✓	✓
Order Restrictions	✓	✓	✓	✓	✓	✓
Fixed Pause Times	✓	(✓)	✓	✓	✓	✓
Heterogeneous Fleet	✓	✗	✗	✓	✓	✓
Multiple Depots	✓	✗	✓	✓	✓	✓
Pickup/Delivery	✓	✗	(✓)	✓	✓	✓
Stop Options	✓	✗	✗	✓	✓	✓
Allow Return	✓	✗	✗	✓	✓	✓

scenario. Since this complete and optimal algorithm requires high computation time, it is only applied to the smallest test instance. The second algorithm is based on a Savings algorithm [CW64] that can handle cases with a homogeneous vehicle fleet, a single depot, and no pickup and delivery problem. Even if we know on which approach this algorithm is based, we call it Blackbox-I as we have no insight into the details of the implementation. The third algorithm (Blackbox-II) is an extension to the above mentioned Blackbox-I algorithm covering a multi-depot problem and more complex pause time rules. Both Blackbox algorithms are proprietary algorithms developed by our cooperation partner. The fourth algorithm supports all features required for our rVRP as it uses our model of the problem inside OptaPlanner and is an implementation of Tabu Search [Glo86] provided by the OptaPlanner's LS algorithms.

Since we modeled the rVRP inside OptaPlanner additional optimization could be applied such as exhaustive search, hyper-heuristics or partitioned search. However, we decided to use the Tabu Search implementation as promising representative of LS algorithms. Further, other optimization techniques could be applied on the rVRP such as exact algorithms by using an adjusted penalty function. However, several restrictions of our problem statement prevented us from using these as for example we retrieve the stop-to-stop distance from a service of our cooperation partner and this information is not available as fixed adjacency matrix that could be transferred to other algorithms easily. Further, the integration of all handled constraints into one penalty function is problematic since diverse constraints need to be reduced to one value. This single value is not a well fitting indicator which of the constraints is violated and therefore a directed search towards an optimum is hardly possible.

11.3 Evaluation Procedure

We evaluate our proposed approaches against the alternative four algorithms on all problem instances defined in Section 11.1. We execute the probabilistic algorithms (LS, GA, ACO) 30 times with different random seeds to retrieve representative results for comparison. We summarize the parametrization of our GA in Table 11.3. For defining the population size of the genetic algorithm, we use statistics of the problem instance to be solved. Therefore, we

Table 11.3: Parametrization of our proposed GA approach.

Parameter	Value
Population size	$0.1 \cdot \#o + \#v^{1.25} + \#(\text{PD-orders}) + 2 \cdot \#(\text{multi-option stops})$
Mutation probability (p_{vrp}, p_{tsp})	0.5
Max. unimproved iterations	500

use 10% of the number of orders, add the number of vehicles to the power of 1.25, add the number of pickup delivery orders and twice the number of stops containing multiple options. We derived these values in a preliminary parameter study and focused on providing a reasonable number of individuals regarding the complexity of the problem. We set the mutation probability to be 50%, hence, on average half of the newly created individuals are mutated and set the termination criterion to a maximum iterations without improvement to be 500. For the ACO we use an evaporation factor of 0.05, the size of the set of best solutions so far to ten, and the maximum number of iterations without improvement to 500. For the evaluation runs, we used a server exclusively for our measurements with the following specifications: Two Intel(R)Xeon(R) CPU E5-2667 v4 processors with 3,20 GHz each with 16 GB of RAM. Windows Server 2012 R2 Datacenter runs as a 64-bit operating system on the server.

Table 11.4: Parametrization of our proposed ACO approach.

Parameter	Value
Evaporation factor	0.05
Size set of best solutions (N)	10
Max. unimproved iterations	500

11.4 Results and Interpretation

In the following we discuss the results of the algorithms on all defined problem instances. Since we use the ranked score for measuring the quality of the solutions, all hard scores need to be reduced to zero to consider a solution feasible. In case the hard scores (H_1, H_2, H_3) are not down to zero, the algorithm does not find a feasible solution, which we indicate with dashes (-) in our results table (Table 11.5). Further, the soft scores aim at the matched time windows in the first soft score (S_1) and the tour length in the second soft score (S_2) that both need to be minimized. The optimization considers the third soft score (S_3) only if the previous scores are reduced to zero. As this is not the case in any of our evaluation results, we do not discuss this score in our evaluation. However, it is important to have a score definition for the addressed objectives as it could be required as tie breaker if previous soft scores are exactly the same. Keep in mind, that even if pause times and time windows are handled in the Timeline algorithm, no pause time violations can occur as ensured by our algorithm and we only include the time window violations in the score S_1. For better readability, we re-scaled all values by dividing them by 10,000 in our result presentation. To sum up the evaluation results of all problem instances, we provide Table 11.5 that states whether time windows are met (S_1) as well as mean and standard deviations of the tour length (S_2) over 30 runs for probabilistic algorithms, that is, the LS, GA, and ACO. Ticks (✓) indicate, that all time windows are met in all runs of the algorithm, crosses (✗) show that these are not met. A value of 19/30 for the S_1 score shows that in 19 from 30 runs, all time windows are met. We report the results of the brute force algorithm exclusively for the TSP-I problem instance since it already took the algorithm 7 hours and 15 minutes to find a solution for this problem instance. For larger problem instances, for example the TSP-II problem instance, the algorithm has to assess $33! = 8,6 \cdot 10^{36}$ possible solutions of sequences and, hence, was not able to calculate the optimum solution within feasible time. Further, we consider a maximum calculation time for all algorithms which we examined in cooperation with our partner to have representative real-world results. We define this maximum calculation time to stay within a practical applicable runtime of the algorithms between 60 and 300 seconds. We decided to set these time limits as we want to be able to react to changes in the orders, vehicles, and stops at any point in time. Hence, we do not aim at planning the rVRP once at the beginning of the day, contrary we aim to be adjustable at any time to represent adaptive behavior using feedback loops. We calculate the mean and standard deviation values in the table using the final score values of the solutions provided after the execution time. We test both Blackbox algorithms for all problem instances except for the pickup and delivery instances since they are not designed to handle pickup and delivery problems. Additionally, we provide line charts and box plots for all problem instances. The line charts represent the course of the mean values over 30 repetitions of the S_2 score throughout the optimization. For non-deterministic algorithms (LS, GA, ACO) we further show the standard deviations as error bars. The box plots represent the final S_2 results of the algorithms after the execution time is over. To make statements on statistical significance of the results we perform Wilcoxon signed rank tests for the non-deterministic algorithms in all relevant comparisons. We define the null hypotheses to be that the mean values are drawn from the same distribution and, hence, have no statistical significant difference. Further, we define the significance level to be $\alpha = 0.05$. In the following, we first present the results for the TSP instances, than the ones for the VRP instances and finally discuss our findings of the P&D instances.

11.4.1 TSP-I

The table shows that for the TSP-I problem instance, the LS and GA are able to fulfill all time windows and find the best possible score value (determined by the result of the brute force algorithm). The Blackbox-II algorithm finds a solution with a reduced score value of around 2.6 score points less but was not able to fit the time windows. The Blackbox-I and ACO algorithms are able to fit all time windows but they find only solutions with higher score value, that is, around 1.00 and 2.35 score points above the optimal score value, respectively.

Table 11.5: Summary of the evaluation results for S_1 (time windows = TW) and S_2 (tour length score) for all algorithms and problem instances. For probabilistic algorithms, the mean and standard deviation values over 30 runs are listed (P = with pause times, P&D = with pickup and delivery). From all solutions that were able to match all time windows represented in the first soft score the best values for the second soft score are shown in bold.

Algorithm	**Brute Force**		**Blackbox-I**		**Blackbox-II**		**LS**			**GA**			**ACO**		
	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2		S_1	S_2		S_1	S_2	
								mean	std		mean	std		mean	std
TSP-I	✓	**91.52**	✓	92.55	✗	90.95	✓	**91.52**	0	✓	**91.52**	0	✓	93.87	2.06
TSP-II	-		✓	161.87	-		✓	157.62	5.68	✓	**156.74**	0.89	✗	222.00	7.50
TSP-II-P	-		✗	161.87	-		19/30	207.41	30.50	25/30	190.53	20.26	✗	217.74	7.58
VRP-I	-		✓	187.14	✓	185.71	✓	186.56	8.25	✓	**177.19**	1.22	✓	201.81	8.85
VRP-I-P	-		✗	187.14	✗	177.82	✓	194.10	5.84	✓	**179.81**	0.67		-	
VRP-II	-		✓	396.21	✓	373.05	28/30	299.03	60.95	✓	**292.86**	14.60		-	
TSP-PD	-		-		-		21/30	108.50	2.00	27/30	**104.89**	17.76		-	
VRP-PD	-		-		-		✓	337.80	18.40	✓	**332.39**	13.96		-	

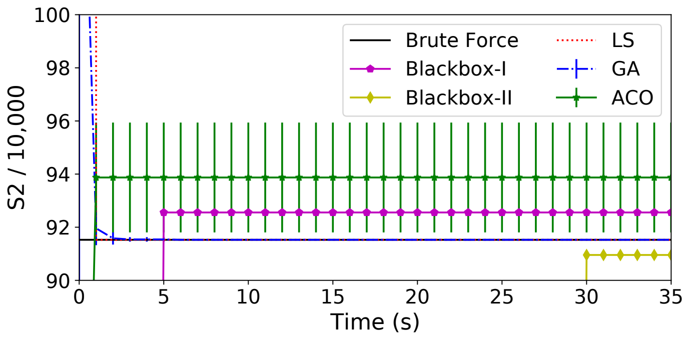


Figure 11.1: Mean and standard deviations of the tour length score (S_2) to be minimized for the TSP-I problem instance for all algorithms in the course of their execution. Mean and standard deviations are calculated for the algorithms LS, GA, and ACO.

Figure 11.1 shows the mean and standard deviation values of the S_2 score for all algorithms during the course of optimization. We limit the depicted time scale to 35 seconds for better identification of differences in the early stage of computation between zero and five seconds. The x-axis shows execution time in seconds while the y-axis presents the S_2 score value divided by 10.000 to achieve better visibility of the values. The Brute Force algorithm is depicted as constant black line at 91 score points for better comparability with the other algorithms even if it took more than seven hours to return the result. Both Blackbox algorithms (depicted in purple and yellow) do not provide the possibility to show the course of optimization but provide a final result after their calculation. The result of the Blackbox-I depicted in purple is returned after five seconds with a higher value of 92.55 than the optimal one of 91.52, while the Blackbox-II algorithm depicted in yellow requires 30 seconds calculation time and returns a lower score of 90.95 while failing to match the time windows. The LS in red and GA in blue show very fast convergence towards the optimum solution in all repetitions, and, hence, show small standard deviations. Contrary, the ACO algorithm depicted in green is not able to achieve the best solution and shows comparably high standard deviations of two score points. We analyzed this behavior of the ACO to identify possible issues that prevent the algorithm from better and more stable results. One possible issue could be the use of a priority score system with three hard and three soft scores to cover the multi-objectiveness of the problem statement. The design of the ACO, however, requires to reduce this score to a single pheromone matrix to guide the optimization process. Hence, we first decided to focus the pheromone matrix on the S_1 score, that is, the time window compliance. However, this

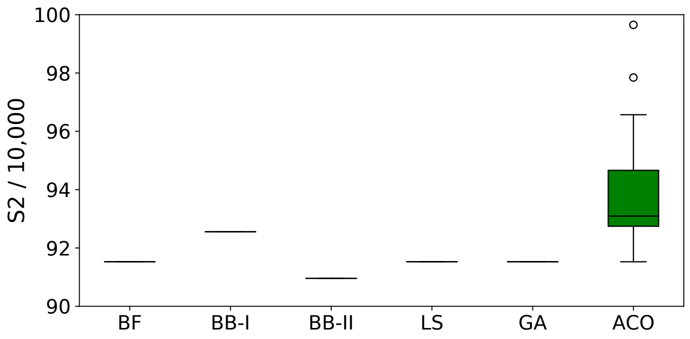


Figure 11.2: Box plot of the tour length score (S_2) to be minimized for the TSP-I problem instance for all algorithms. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time.

resulted in solutions with matched time windows but non-competitive tour lengths as the algorithm was not able to turn the focus of the pheromone matrix on the S_2 score after fulfilling all time windows to optimize the tour length. Then, we experimented on the current status of the ACO and its pheromone handling to always compare the current solution with the global worst score to handle this problem. However, as the results of this evaluation show, this approach was not able to fix both problems. Hence, we identified the need for further research in the future on this problem to address a multi-object priority score when applying ACO algorithms. This comparably bad performance of the ACO can also be observed in the box plot in Figure 11.2 where the ACO is the only algorithm that shows a large box and even outlier with up to 100 score points. Again, it can be seen that the Blackbox-I is not able to compute the optimal solution while the Blackbox-II algorithm is not able to match all time windows and cannot compete with the other algorithms. Since LS and GA computed the optimal solutions in all repetitions without any deviations, we did not perform statistical tests on this problem instance. In summary, LS and GA were able to achieve the best possible solution after only a few seconds and in all runs while the Blackbox algorithms produce worse solutions and the ACO cannot compete with the other algorithms.

11.4.2 TSP-II

In the following, we analyze the larger TSP instance consisting of 30 orders, one vehicle without P&D and pause times. Table 11.5 shows that for the TSP-II instance, the Brute Force algorithm was not able to calculate the optimal

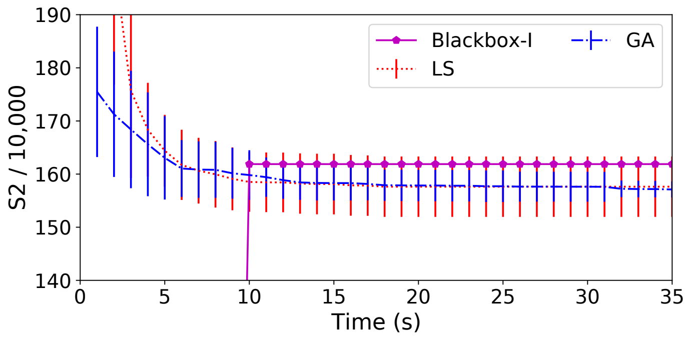


Figure 11.3: Mean and standard deviations of the tour length score (S_2) to be minimized for the TSP-II problem instance for all algorithms in the course of their execution. Mean and standard deviations are calculated for the algorithms LS, and GA.

solution, while Blackbox-II and ACO were not able to match the time windows. The Blackbox-I, LS, and GA find solutions that match all time windows and comparable S_2 score values of around 160 score points with GA showing the lowest score with 156.74. Figure 11.3 shows the mean and standard deviation values of the S_2 score for the algorithms that were able to match all time windows during the course of optimization, that is, Blackbox-I, LS, and GA depicted in purple, red, and blue, respectively. Again, we limit the depicted time scale to 35 seconds for better identification of differences in the early stage of computation between zero and ten seconds. While the LS starts at a comparably high score value above the depicted scale, the GA starts with an already good value of around 170 score points. Both algorithms are able to drastically reduce the mean and standard deviation score values to around 160 score points after ten seconds of execution time. The Blackbox-I again delivers its solution of 161.87 after it finishes its calculation after ten seconds.

In the course of the execution, the LS and GA alternate with the best value, however, remain at a comparable level especially considering the standard deviation. Figure 11.4 shows the results for the three algorithms as box plot calculated using their final score values after 30 repetitions. The result of the previous evaluation is also reflected in this plot as the Blackbox-I shows a higher score than the mean scores of the LS and GA. The LS shows the larger box and, hence, a wider variety of solution with outliers that even range up to a score value of around 180. In contrast, the GA shows stable behavior with a slightly lower mean value with a difference of 0.9 score points compared to the LS. We performed Wilcoxon signed rank tests to check for statistical significance in the test results between the non-deterministic LS and GA. We

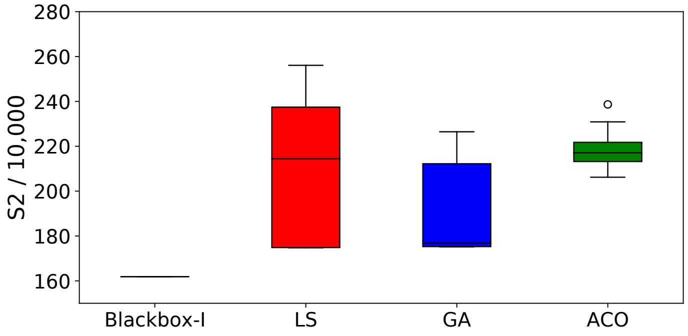


Figure 11.4: Box plot of the tour length score (S_2) to be minimized for the TSP-II problem instance for all algorithms that matched the time windows. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time was reached.

define the H_0 hypotheses to be that the mean values are drawn from the same distribution and calculated a p-value of 0.185. Hence, we were not able to reject our hypotheses with a significance level of $\alpha = 0.05$. In summary, the LS and GA calculate best solutions after ten seconds and perform equally well.

11.4.3 TSP-II-P

When looking at the next problem instance, we evaluate the performance of the algorithms using the TSP-II instance and additionally include pause times. As can be seen in the overview in Table 11.5 the Blackbox-I, Blackbox-II, and ACO algorithm were not able to match all time windows in any of the proposed solutions. Contrary, the LS was able to match time windows in 19 from 30 solutions and the GA in 25 of 30 solutions. Additionally, the GA produces results with lower S_2 score of 190 score points compared to 207 score points (LS) and lower standard deviation (20 score points for GA and 30 score points for LS). Again, Figure 11.5 provides the mean and standard deviation values of the S_2 score for the Blackbox-I, LS, GA, and ACO algorithms during the course of execution. We limit the depicted time scale to 35 seconds for better identification of differences in the early stage of computation between zero and ten seconds. Since no algorithm was able to match all time windows in all runs, we present the results of all algorithms returning a value for S_2 and keep the performance regarding the time windows in mind. The Blackbox-I algorithm depicted in purple returns its solution of 161 score points after ten seconds and has a lower S_2 score compared to the other algorithms. However, as this solution does not match any time window, we consider it worse than the other

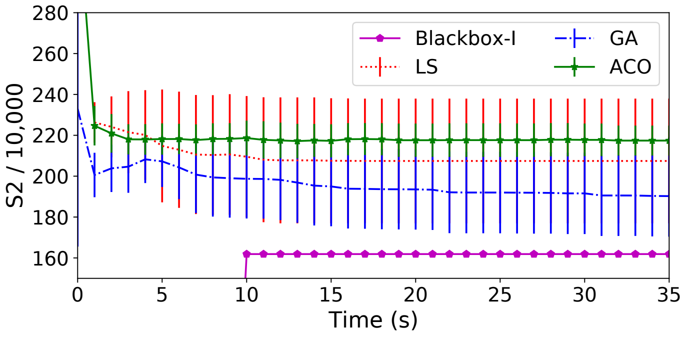


Figure 11.5: Mean and standard deviations of the tour length score (S_2) to be minimized for the TSP-II-P problem instance for all algorithms that returned a result for S_2 in the course of their execution. Mean and standard deviations are calculated for the algorithms LS, GA, and ACO.

algorithms. The course of optimization of the LS depicted in red, GA in blue, and ACO in green show, that the GA already starts with a better value (230 score points) than both other algorithms (higher than 280 score points) and continues to decrease the score slightly during runtime. However, we observe a small increase in mean score points after four seconds which might be due to our diverse set of crossover and mutation operators that might decrease the overall quality of the solutions in trade for a higher diversity of the population. Still, from five seconds of execution time onward, the GA reduces the mean score value continuously. The LS is also able to decrease its score but a high standard deviation of 30 score points compared to 20 score points for the GA can be observed while the ACO seems to show no improvement at all with a very low standard deviation of 8 score points. This can be explained by the fact that the ACO was not able to match the time windows in any run and hence, does not focus on optimizing the S_2 score. The box plots in Figure 11.6 show similar results with a high mean value for the LS and the ACO and a low mean for the GA. As the ACO was not able to match the time windows in any run, we consider its performance worse than the ones from LS and GA. After the execution time, the GA shows a lower score compared to the LS which we check for significance using a statistical test. We again performed a Wilcoxon signed rank test to compare LS and GA and calculated a p-value of 0.082 and were not able to reject our hypotheses with a significance level of $\alpha = 0.05$. This means, that we cannot state that the calculated mean values are drawn from distinct distribution and no statement can be made regarding statistical significance of the result differences. In summary, all algorithms were not able

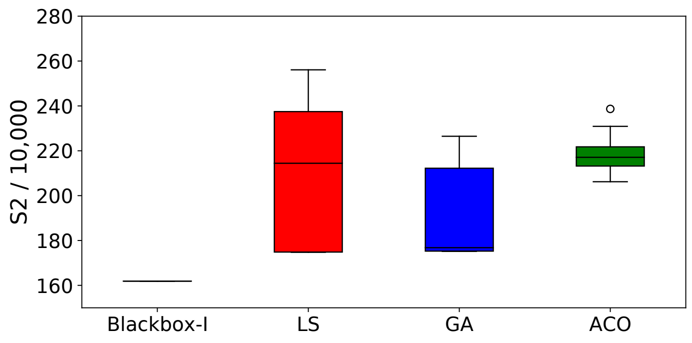


Figure 11.6: Box plot of the tour length score (S_2) to be minimized for the TSP-II-P problem instance for all algorithms that returned a result for S_2. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time was reached.

to find solutions with matching time windows in all repetitions. However, the LS and GA were able to match time windows in some of the repetitions and, hence, we consider them as best performing in this problem instance.

11.4.4 VRP-I

We now analyze the results of the algorithms in the VRP-I instance that addresses both components of the vertical systems-of-systems and consists of 53 orders that need to be scheduled using five vehicles. The results using the first VRP problem instance (VRP-I) summarized in Table 11.5 show, that all tested algorithms are able to match the time windows. While the ACO provides solutions with a high S_2 score of around 200 score points, the scores of the other algorithms are comparable low around 186 score points with the GA showing the lowest value of 177 score points. The line chart in Figure 11.7 shows the course of optimization for all algorithms during the execution time. We limit the depicted time scale to 120 seconds for better identification of differences in the early stage of computation in the first 60 seconds. The Blackbox-I algorithm delivers its result of 187 score points after around 18 seconds while the Blackbox-II algorithm requires 65 seconds calculation time with a score value of 186 score points. Both algorithms deliver results with higher S_2 score value compared to the GA with the lowest score of 177. The GA already starts with a good initialized value of around 185 and continues reducing the S_2 score over time with small standard deviations of around 1 score point. The LS algorithm starts with a high mean value outside the depicted scale but reduces the score to the level of the Blackbox algorithms in the first 20 seconds but is not able to

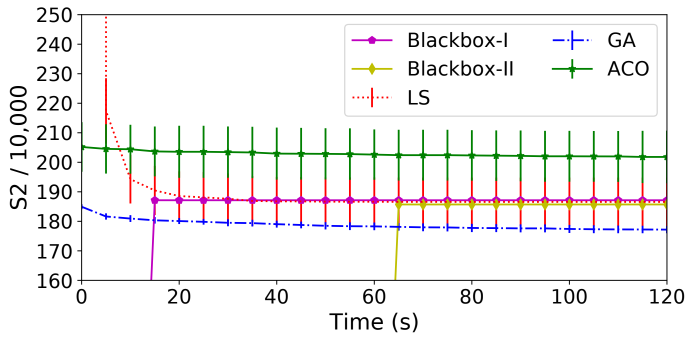


Figure 11.7: Mean and standard deviations of the tour length score (S_2) to be minimized for the VRP-I problem instance for all algorithms that returned a result for S_2 in the course of their execution. Mean and standard deviations are calculated for the algorithms LS, GA, and ACO.

reach the level of the GA and shows larger standard deviations of around 8 score points. The ACO algorithm shows higher score values of around 208 score points compared to the other algorithms but slightly reduces the score over time with standard deviations of around 6 score points. A similar result can be seen in Figure 11.8 where the GA shows the lowest values and the smallest box indicating a very stable low score value. The mean of the LS is similar to the values for the Blackbox algorithms but has a larger box and hence, shows a larger diversity in the results and an outlier that reaches up to 220 score points. Finally, the ACO has a higher mean value and a larger variety in the results that spans from 190 to 220 score points. As the ACO algorithm was not able to compete with the other algorithms in the previous problem instances due to the discussed issues in relation to the multi-objective problem statement, we omit to report results for this algorithm in the even more complex scenarios following in the next section. To check the statistical significance of these results, we again perform the Wilcoxon singed rank test. Since this test assesses pairs of algorithms to be compared, we apply it pairwise on all possible combinations between LS, GA, and ACO. Using this test we are able to reject the hypotheses with p-values of 0.001 and a significance level of $\alpha = 0.05$, which means that the mean values are drawn from different distributions and hence the difference between all algorithms is statistically significant. In summary, the GA shows significant improvements over the LS and ACO algorithms in this instance.

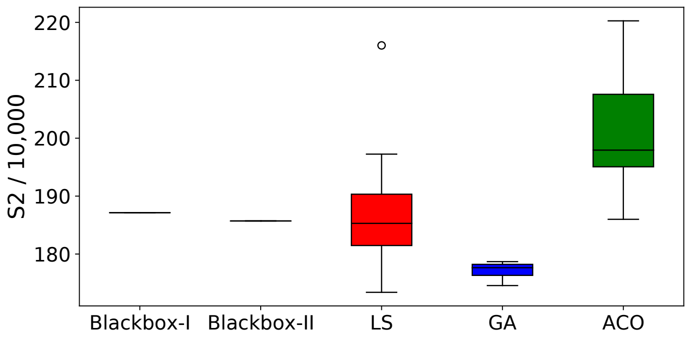


Figure 11.8: Box plot of the tour length score (S_2) to be minimized for the VRP-I problem instance for all algorithms that returned a result for S_2. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time was reached.

11.4.5 VRP-I-P

The problem instance we evaluate in the following bases on the previous VRP-I problem instance and adds pause times to the problem statement. Table 11.5 shows for the VRP-I-P problem instance with pause times that both Blackbox algorithms are not able to match the time windows. Contrary, the LS and GA solutions match all time windows with GA having the lowest mean of 180 score points and standard deviation of 0.7 score points. The course of the optimization is depicted in Figure 11.9 for all algorithms regardless whether they matched all time windows. We limit the depicted time scale to 120 seconds for better identification of differences in the early stage of computation in the first 80 seconds. However, we keep in mind that both Blackbox algorithms did not match the time windows. The figure shows that the Blackbox-I delivers its result of 187 score points after around 18 seconds while Blackbox-II algorithm requires 80 seconds calculation time with a final score of 177. Again, the GA starts with a already good solution of around 185 score points and further reduces the score value throughout the calculation time while the LS starts with a very high score larger than 250 score points. Additionally, the GA shows a very small standard deviation and, hence, produces very stable results while the LS shows comparably large standard deviations. The LS is able to reduce its score within the first 30 seconds but still has a difference of around 15 score points to the GA. The box plot in Figure 11.10 supports this finding, as the GA has a low score value and produces very stable results with only a few variations, while the LS shows worse results and high variability in the solution quality ranging from 187 to 207 score points. The Blackbox-I algorithm

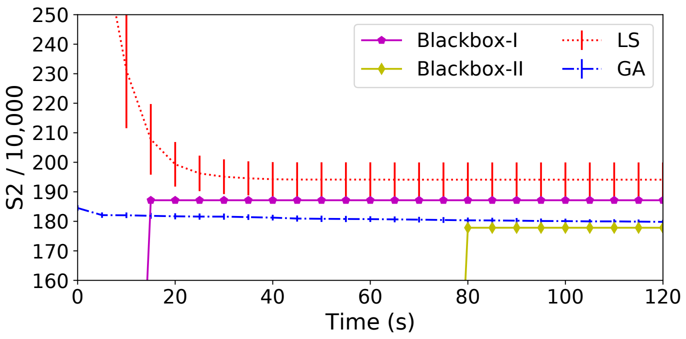


Figure 11.9: Mean and standard deviations of the tour length score (S_2) to be minimized for the VRP-I-P problem instance for all algorithms that returned a result for S_2 in the course of their execution. Mean and standard deviations are calculated for the algorithms LS and GA.

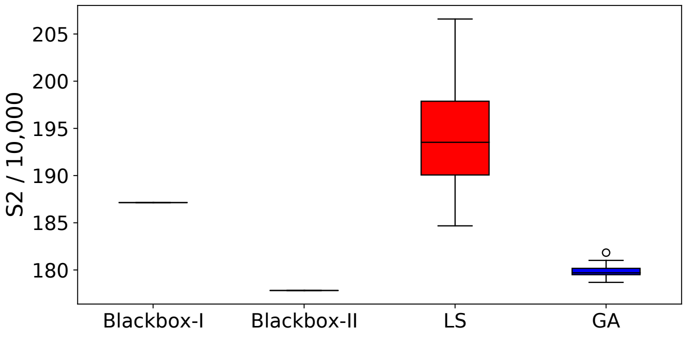


Figure 11.10: Box plot of the tour length score (S_2) to be minimized for the VRP-I-P problem instance for all algorithms that returned a result for S_2. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time was reached.

shows a larger value compared to the GA and a lower mean value compared to LS while the Blackbox-II algorithm shows a lower score value than the other algorithms. Still, we consider both Blackbox algorithms worse than LS and GA as they are not able to match time windows in this problem instance. We again performed a Wilcoxon signed rank test between LS and GA to analyze the statistical significance of the results. The test calculates a p-value of 0.001 and, hence, we are able to reject the hypotheses with a significance level of $\alpha = 0.05$ what means that the values are statistically significant different. In summary, the GA shows significant improvements over the LS and shows the most stable solution quality.

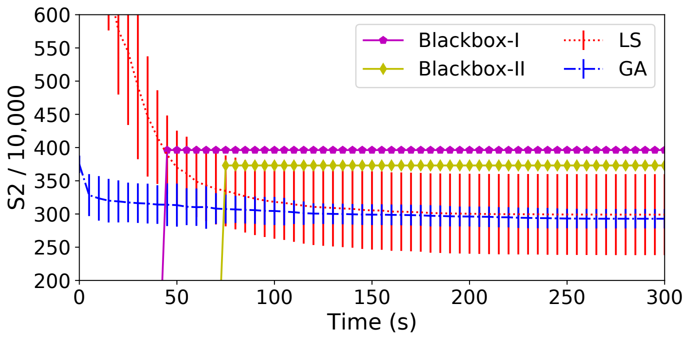


Figure 11.11: Mean and standard deviations of the tour length score (S_2) to be minimized for the VRP-II problem instance for all algorithms that returned a result for S_2 in the course of their execution. Mean and standard deviations are calculated for the algorithms LS and GA.

11.4.6 VRP-II

The problem instance VRP-II is a larger version of the VRP-I instance and consists of 100 orders that need to be scheduled to 13 vehicles. This problem instance does not include pause times. As the summary in Table 11.5 show that both Blackbox algorithms are able to match all time windows, while LS only matches time windows in 28 of 30 runs. The solutions of GA match all time windows in all runs, and hence, are considered better than the solutions of LS. Figure 11.11 shows the course of optimization during calculation time for all algorithms. The Blackbox-I algorithm returns its result of 396 score points after around 45 seconds, while the Blackbox-II algorithm delivers its solution after around 75 seconds with a score of 373. The GA starts with a solution quality in the area of the Blackbox algorithms and further reduces the score value to 293 and its standard deviation to 15 score points over time. In contrast, the LS starts with a high score outside the depicted scale and reduces its solution drastically to the level of the GA at around 100 seconds but shows a comparably high standard deviation of 61 score points. However, LS and GA show considerably lower values compared to both Blackbox algorithms. The box plots in Figure 11.12 also shows this finding that LS and GA have lower mean values than the Blackbox algorithms. However, the LS shows a broader variety in the solution quality with whiskers ranging from 250 to 350 and an outlier at around 580 score points while the GA's whiskers range from 280 to 310 and shows an outlier at around 350 score points. Hence, the solution quality and the variety of the quality of the final results of both algorithms is comparably good with a small advantage for the GA. We performed a Wilcoxon

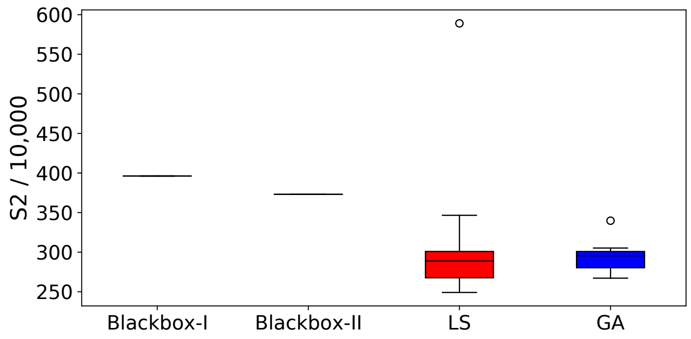


Figure 11.12: Box plot of the tour length score (S_2) to be minimized for the VRP-II problem instance for all algorithms that returned a result for S_2. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time was reached.

signed rank test to compare the results from LS and GA which calculates a p-value of 0.478. Hence, we are not able to reject our hypotheses and cannot state that the mean values are drawn from distinct distributions. In summary, the LS and GA algorithms outperform both Blackbox algorithms with regards to the second score and perform comparably good. However, the LS does not match the time windows in all runs and is considered worse than the GA.

11.4.7 TSP-PD

For both P&D problem instance, we compare the LS and the GA since the Blackbox algorithms cannot handle P&D problems. We first evaluate the algorithms using a TSP-PD instance that consists of 10 orders to be scheduled on one vehicle with P&D requirements but without pause times. The results in Table 11.5 show for the TSP-PD problem instance that the LS matches time windows in 21 from 30 runs, and the GA in 27 from 30 runs. Hence, the GA can be considered more stable than the LS as the probability to receive solutions with matching time windows is higher. The LS produces a mean score value of 108 score points with a standard deviation of 2 score points while the GA has a mean value of 105 and a standard deviation of 18 score points. Figure 11.13 shows the optimization result during the runtime of the algorithms with a maximum computation time of 60 seconds. Both algorithms start with high score values above the depicted scale and reduce the score in the first 2-3 seconds to values of around 108 for the LS and 105 for the GA. The GA shows larger standard deviations of around 18 score points compared to the LS with a standard deviation of 2. This can also be seen in Figure 11.14 where the box

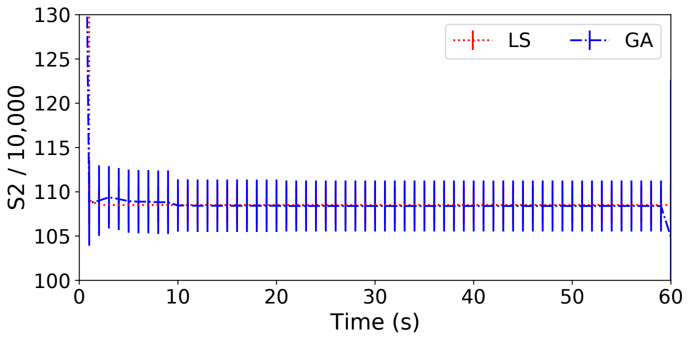


Figure 11.13: Mean and standard deviations of the tour length score (S_2) to be minimized for the TSP-PD problem instance for all algorithms that returned a result for S_2 in the course of their execution. Mean and standard deviations are calculated for the algorithms LS and GA.

plot of the LS is very small besides one outlier at around 117 score points and the GA box plot shows a broader range from 108 to 110 score points. We again performed a statistical test to analyze the statistical significance of our results. The Wilcoxon signed rank test was not able to reject the hypotheses with a p-value of 0.145 and hence, we cannot state that both mean values are drawn from distinct distributions. In summary, both algorithms perform comparably good in this problem instance as both do not match all time windows and deliver nearly the same quality in the S_2 score.

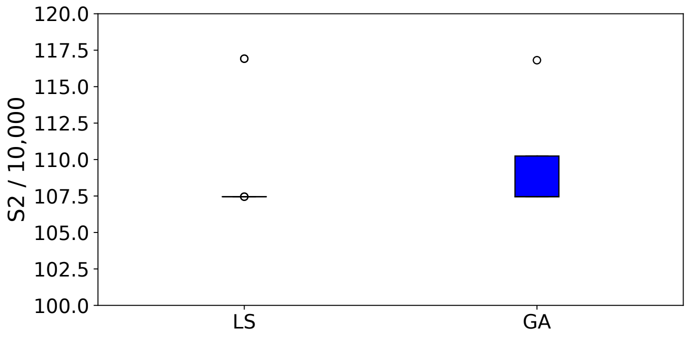


Figure 11.14: Box plot of the tour length score (S_2) to be minimized for the TSP-PD problem instance for all algorithms that returned a result for S_2. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time was reached.

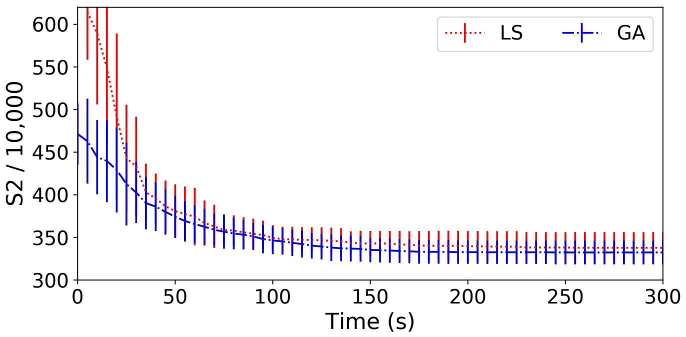


Figure 11.15: Mean and standard deviations of the tour length score (S_2) to be minimized for the VRP-PD problem instance for all algorithms that returned a result for S_2 in the course of their execution. Mean and standard deviations are calculated for the algorithms LS and GA.

11.4.8 VRP-PD

Finally, we analyze our algorithms on a VRP-PD instance that includes 63 orders that need to be scheduled using seven vehicles and that integrates P&D as well as pause times. Again, Table 11.5 summarizes the results and shows that LS and GA are able to match all time windows in the VRP-PD problem instance. Further, the GA produces solutions with a lower mean S_2 score value of around 332 score points compared to the LS with a value of 338. The standard deviations of both algorithms are comparably low with values of 18 and 14 for the LS and the GA, respectively. In Figure 11.15 both algorithms start with a high value of 450 for the GA and 600 score points for the LS but decrease the score in the first 100 seconds to around 350 score points. Still, the GA maintains its lead and its mean value stays below the mean of the LS. The box plots in Figure 11.16 show that the final mean values are also very similar. The box and whiskers of the LS span a wider range from 300 to 355, while the GA has a smaller box ranging from 325 to 360 but some more outliers below and above the whiskers. We performed a statistical test to analyze the statistical significance of our results. The Wilcoxon signed rank test was not able to reject the hypotheses with a p-value of 0.329 and, hence, we cannot state that the derived mean values are drawn from distinct distributions. In summary, again both algorithms perform equally good with a slight advantage of around 50 score points for the GA.

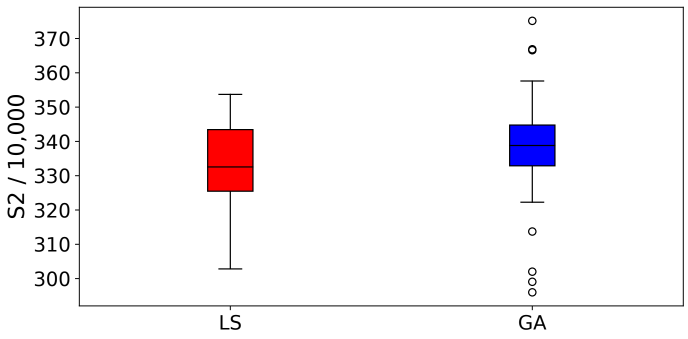


Figure 11.16: Box plot of the tour length score (S_2) to be minimized for the VRP-PD problem instance for all algorithms that returned a result for S_2. The box plot is calculated with the final solutions the algorithms return after their execution finished or the maximum computation time was reached.

11.5 Threats to Validity

We identified the following threats to validity for our approach. In this paper, we focus on nature-inspired algorithms (ACO and GA) for tackling the rVRP and compared them to a Brute-Force, two Blackbox algorithms implemented by our cooperation company, and LS. Those algorithms provide heuristic solutions, which provide fast results, however, require multiple runs to receive reliable results. Further, we did not evaluate other common algorithms used for these kinds of problems as those often require manual implementation effort to adjust them for the particular rVRP problem. Therefore, we decided to compare our algorithms to an existing implementation of LS inside OptaPlanner. Additionally, our results are limited to the defined problem instances and we plan to also evaluate even larger VRP instances in cooperation with our cooperation company in the future. Finally, our analysis of related work showed that existing approaches simplify the problem by using assumptions or neglecting specific aspects. One could argue that we over-complicated the problem as so far it has been enough for the industry to solve the trimmed-down versions. However, as the problem formulation was motivated and done with our cooperation company, these constraints reflect an actual need from practice. Further, we think that in the course of digitization in industry, industry will be faced with increasing complex problems and solving them in an automated way without limitations might be a competitive advantage.

11.6 Summary

In this chapter, we evaluated our first contribution, which focuses on **Goal B**: *Improving the quality of optimization strategies in complex systems-of-systems with a special attention to the field of logistics*. We answer **RQ B.1** and its subordinate question **RQ B.1.3** by analyzing the proposed approaches on the use case rVRP. The evaluation showed that the approach integrates all real-world constraints and reduces the time-to-result from 30 to 2 seconds and from 60 to 5 seconds for the TSP and VRP instances, respectively. These results have an tremendous practical benefits and show that the proposed workflow takes into account the complex systems-of-systems structure.

Chapter 12

Horizontal Systems-of-Systems Approach

In this chapter, we evaluate our approaches for the horizontal systems-of-systems problem statement of the logistics domain: the storage assignment and order picking in a mezzanine warehouse. We introduced our approach in Chapter 9 and proposed an NSGA-II for the storage assignment and an ACO for order picking. We analyze the performance of our algorithms on three different warehouse models representing different sizes of warehouses. For all three warehouse models, we execute our algorithms and compare them with policies commonly used in mezzanine warehouses. To analyze the performance, we use several performance indicators to judge the quality of Pareto fronts as introduced by [WAY$^+$16] and Section 3.3. Therefore, we first evaluate the storage assignment task for all warehouse models, followed by the order picking, and finally, we evaluate the interaction between the storage assignment and order picking approach.

This chapter is based on our technical report [LMK$^+$21b] and our paper under review in the Springer Applied Intelligence Journal [LMK$^+$21a] and is structured as follows: Section 12.1 introduces our three different warehouse models including the fill strategy and properties of the stored products. Section 12.2 presents storage assignment and order picking strategies often used in real world mezzanine warehouses to which we compare our approaches. Section 12.3 summarizes the parameter settings for our approaches that we used in our evaluation on the three warehouse models. Section 12.4 presents and discusses our results of our storage assignment task, Section 12.5 summarizes the results of the order picking, and, Section 12.6 discusses the interaction of both warehouse tasks. Afterward, Section 12.7 analyzes threats to validity and, finally, Section 12.8 summarizes this chapter.

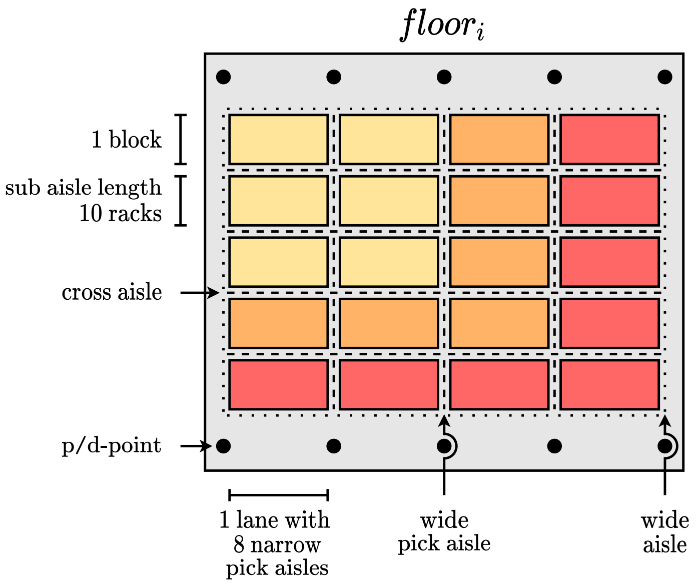
12.1 Mezzanine Warehouse Models

To evaluate our proposed algorithms, we aim at a realistic scenario to obtain results as realistic as possible. Therefore, we cooperated with a consulting company that provided us with real-world example warehouse layouts, product assortments, and customer orders. Unfortunately, we are not able to publish this dataset as it is subject to a non-disclosure agreement. Based on this data, we design three artificial mezzanine warehouses of different sizes shown in Figure 12.1: WH_{small} (yellow), WH_{medium} (orange), and WH_{large} (red).

All warehouses consist of two identical floors and apply a similar layout as depicted in the figure: The left side depicts the warehouse layouts, the different colors represent the three warehouse sizes, while the right side provides information about the number of floors, racks, and markets for each warehouse. The warehouse layouts on the left show black dots that represent P&D points. The colored rectangles represent one block each that consists of eight narrow pick aisles and 20 racks per sub aisle. The horizontal dotted lines indicate cross aisles, while the vertical lines represent wide pick aisles. The yellow warehouse consists of 6 blocks, 1020 racks, and 8 markets per floor, that is, 12 blocks, 2040 racks, and 16 markets for the overall warehouse. The orange warehouse consists of 12 blocks, 2080 racks, and 15 markets per floor, while the red warehouse consists of 20 blocks, 3500 racks, and 24 markets per floor. All warehouse sizes consist of two identical floors.

For the small, medium, and large warehouses, we define the size of the product assortment to be 500, 1000, and 1500, respectively. Since each product requires a weight, we need to define a process to assign weights to all products. We aim at a representative set of product weights where most of the products have a medium weight and some products have low and some have heavy weights. Hence, we define three normal distributions and a probability to determine the weight using this distribution: 25% to use $\mathcal{N}(2, 1.0)$, 50% to use $\mathcal{N}(5, 2.0)$, and 25% to use $\mathcal{N}(8, 1.0)$. This means, for example, that with a probability of 25% we assign the considered product a weight using $\mathcal{N}(2, 1.0)$, that is, a normal distributed weight with mean of two and a standard deviation of one. Additionally, the products might also have correlations to up to three other products: With a probability of 30%, 40%, 20%, and 10% a product has no, one, two, or three correlated products, respectively, with a randomly generated correlation confidence between 10% and 90%.

For evaluating the order picking algorithm, we fill the storage up to 50% of the available storage space using the above mentioned rules and randomly generate 100 customer orders based on the product assortment and given correlations between products. Each customer order comprises 20 items to be picked that are selected as follows: We split the product assortment into four equally sized groups based on the product rank. With a probability of 40%, 30%, 20%, and 10% an order contains an item of the highest, second highest, third highest,



	WH_{size}	#Floors	#Racks		#Markets	
			per floor	total	per floor	total
🟨	WH_{small}	2	1020	2040	8	16
🟧	WH_{medium}	2	2080	4160	15	30
🟥	WH_{large}	2	3500	7000	24	48

Figure 12.1: The warehouse models WH_{small}, WH_{medium}, and WH_{large} with their floor layout as used in the evaluation depicted at the left. The right side shows the number of floors, racks, and markets per warehouse size.

and lowest rank class, respectively, which ensures that high-ranked products appear more often in customer orders. Hence, orders contain diverse products with individual amounts to be picked.

12.2 Alternative Strategies

To compare the performance of our proposed algorithms for storage assignment and order picking, we use several commonly used strategies for both tasks for comparison. We use the following strategies for the storage assignment problem:

- The **random storage assignment strategy** allocates the incoming items to random racks on the floors in clusters of target quantity size [BIH19].

- In the **closest open location storage assignment strategy**, the strategy selects the storage locations for storing an incoming product, which are usually the racks closest to the P&D-points [DKLDR07].

- The **rank-based storage assignment strategy** assigns fast-moving products close to the P&D-points, while slow-moving products are assigned to racks further away [PS99].

For the order picking problem, we solely apply a modified S-Shape heuristic [Pet97] and constructs s-shaped pick routes based on the graph representation. This heuristic uses all markets as starting point, traverses the remaining markets using an s-like shape, and additionally, computes the reversed versions of each route to generate a Pareto front of possible solutions.

12.3 Algorithm Parameter Settings

Based on a preliminary parameter study, we parameterize our NSGA-II algorithm as follows: We set the mutation probability to 0.95 for all warehouse sizes so that the mutation operators are applied very frequently. Further, we define parameters dependent on the warehouse size (small/medium/large): The parent population size is set to (50/60/70), and the maximum number of generations to (200/250/300). These values increase with the size of the warehouse since the number of possible solutions increases with the warehouse size and we provide the algorithm more exploration possibilities (population size) and more time (number of generations) for optimizing the solutions.

Further, we set the parameters for our ACO algorithm as follows: Using the results of a preliminary parameter study, we define the pheromone factor α to 1.0 and the heuristic factor β to 2.0. We configure the evaporation factor ρ to 0.02 causing the pheromones to evaporate rather slowly, which enables the algorithm to achieve a higher degree of exploration especially in the early stages. We define the min/max values for the pheromone matrices ($\tau_{min/max}$) to be 1 and 25, respectively, set the floor change penalty to 50, and define the allowed weight difference to 3kg. As stopping criterion, we set the maximum number of cataclysms to 3 and hence, the algorithm terminates after it became stuck for the third time. We set the maximum consecutive iterations without improvements to 20, and the maximum iterations to 250 since these values yield the best results w.r.t. the scores.

12.4 Evaluation of the Storage Assignment Approach

We evaluate our NSGA-II algorithm against the random, closest open location, and rank-based storage assignment policies. We apply our NSGA-II approach and all alternative strategies on the three different warehouse sizes which we call Setting 1.a, 1.b, and 1.c for the small, medium, and large warehouse sizes, respectively. For each setting, we generate five random storage assignment tasks, i.e., we select a random product from the product assortment and set the quantity to be assigned to the quantity already existing in the warehouse. We repeat each task of all settings and, hence, the execution of the NSGA-II algorithm, ten times with different random seeds to reduce random effects and present mean and standard deviation values. Unfortunately, the limited number of repetitions does not allow for meaningful statistical significance tests. However, additional runs may be conducted in the future to confirm the results presented. Afterward, we use all generated solutions of all algorithms and strategies to calculate the reference Pareto front required for the performance indicators. Table 12.1 summarizes the mean and standard values for this storage assignment evaluation and we indicate the best mean values per performance indicator using bold font. Further, Figure 12.2 presents box plots for all applied approaches and strategies including all performance indicators on all three warehouse sizes, i.e., settings. A higher value is better for C and PFS, while a lower value is better for GD, ED, GS, and IGD.

We first analyze the results for the small warehouse listed in the first row of Table 12.1 and at the top of Figure 12.2. The coverage (C) results for the small warehouse show that the NSGA-II Pareto front covers about 90% of the reference Pareto front while the other approaches cover only about 9% and 1%. This shows the poor performance of the alternative strategies, which could lead to a discussion about the usefulness of this selection. However, we have deliberately chosen these alternative strategies because they are commonly used in real application scenarios and in many cases no real optimization takes place. Further, the NSGA-II shows the lowest GD and ED mean values of around 0.04 and 16.48, respectively, and, hence, its Pareto front is located closest to the reference front. However, the values of the competing strategies do not show such a strong difference as with the first metric as they lie between 0.95 and 2.14, and between 21.88 and 28.37 for GD and ED, respectively. Regarding the PFS performance indicator, the NSGA-II algorithm finds around 48 solutions per problem instance with a maximum possible value of 50 solutions for the small warehouse size. The other policies construct only 22 to 28 Pareto-optimal solutions while their maximum possible value is defined to be 500 due to our methodology. Further, the box plot of the NSGA-II shows a smaller variability for this metric and, hence, shows the most stable result. Further, the NSGA-II achieves the lowest GS and IGD values of 0.5 and 0.26, respectively, which indicates that the solutions converge well towards the reference Pareto front and offer diverse solutions. Especially for the IGD metric, the other approaches show a large range covered by the outliers ranging up to a value of 8. In summary, the NSGA-II performs best for the small setting as it is able to optimize all performance indicators best compared to the alternative strategies.

Next, we analyze the medium warehouse listed in the second row of Table 12.1 and in Figure 12.2. In this warehouse, the results show similar performances of all approaches applied in this task. Again, the results show that the NSGA-II Pareto front covers approximately 93% of the reference Pareto front. Except for some outliers, the alternative policies struggle to cover the solutions in the reference front with C values of around 0.00 and 0.06. The observed GD and ED values are fairly similar to the values in Setting 1.a. However, the standard deviations of the ED metric increased noticeably, which may be related to the larger search space where the solutions tend to be more spread out. Nevertheless, the Pareto front of the NSGA-II still achieves the lowest GD and ED values of 0.04 and 16.48, indicating that this Pareto front converges best towards the reference Pareto front. Concerning the PFS metric, the NSGA-II algorithm finds about 53 solutions per problem instance, while the alternative policies find approximately less than 20 solutions per problem instance. The GS values of the NSGA-II increased remarkably from 0.5 in the small warehouse to 1.04 in the medium warehouse, which may be due to the larger parent population size and the larger search space that make it difficult for the NSGA-II algorithm to fill the gaps in the Pareto front so that all solutions are evenly distributed. Accordingly, the closest storage assignment strategy produces the best value for the GD metric of the medium warehouse of 0.73. Still, the box plot shows that the 3rd quartile ranges to a value of 1.5 which indicates that the NSGA-II has the potential to produce clearly better values compared to the other strategies. Lastly, the IGD values of the NSGA-II are close to zero with a value of 0.11 compared to values around 2.00 for the other strategies, indicating that NSGA-II represents the entire reference Pareto front in most cases. In summary, we again consider the NSGA-II algorithm to perform best as it optimizes all indicators best with the exception of the GD value.

Finally, we analyze the results for the large warehouse listed in the third row of Table 12.1 and at the bottom of Figure 12.2. Similar to both smaller warehouses, the large warehouse shows comparable results. The Pareto front of the NSGA-II covers about 99% of the reference Pareto front, while the Pareto front of the rank-based strategy covers only 1%. Thus, almost all solutions found by the rank-based strategy are dominated by the solutions found by the NSGA-II algorithm. The mean GD value of the NSGA-II equals zero, indicating that the entire Pareto front is part of the reference Pareto front in almost all cases. The other strategies have higher GD values between 1.4 and 3.11 and, hence, are outperformed by the NSGA-II. Regarding the ED performance, the results of the competing strategies seem to come closer and have a wider variety in the solution quality indicated by the box plots, which we can confirm by analyzing the mean and standard deviations in Table 12.1 Again, the NSGA-II algorithm finds most solutions per problem instance with a value of 64 compared to values between 9.92 to 14.24 for the other strategies. Similar to the medium warehouse task, the GS values show that the NSGA-II performs worse than the other strategies with a mean value of 1.37 compared to 0.81 to 0.94 for the other strategies. Lastly, the mean IGD value of the NSGA-II of 0.02 indicates that this algorithm outperforms the commonly used strategies for storage assignment with values between 1.86 to 2.10, that, additionally, show very large box plots with whiskers ranging up to a value of 9.

In summary, the results show that the random and the closest open location strategy struggle to cover a single solution in the reference Pareto front while the rank-based strategy at least produces a value of 0.09. Regarding the convergence performance indicators GD and ED, the NSGA-II algorithm produces the best Pareto fronts closest to the reference front compared to the other strategies. For the diversity performance indicators PFS and GS, the NSGA-II always produces the highest number of solutions but gets outperformed regarding the spread of the solutions for the larger warehouse sizes. Finally, the NSGA-II outperforms the other strategies with regards to the IGD, and hence computes the Pareto fronts closest to the reference front. In conclusion, we consider the NSGA-II algorithm best for all sizes and performance indicators, as it outperforms the alternative policies in all except one indicator categories.

In addition, we also measure the mean execution time of the approaches for solving 50 problem instances in each warehouse size. We run our experiments on a MacBook Pro using macOS Sierra 10.12.6, a 2.2GHz Intel Core i7 CPU, and 16GB DDR3 RAM. The alternative policies achieve low execution times of about 0.17/0.30/0.50 seconds for small/medium/large warehouses, respectively, which is due to their simple operation. In the warehouse small/medium/large, the NSGA-II algorithm achieves execution times of about 2/6/15 seconds, respectively, which is due to the increased population and iteration count for larger warehouses. The execution times of the NSGA-II algorithm may be considered acceptable, as the algorithm requires only a few seconds to find remarkably better solutions compared to the alternative policies.

Table 12.1: Mean and standard deviation values of the six quality indicators Coverage (C), Generational Distance (GD), Euclidean Distance (ED), Pareto Front Size (PFS), Generational Distance (GD), and Inverted Generational Distance (IGD) achieved by the storage assignment strategies in Setting 1.a, 1.b, and 1.c (best mean values are shown in bold).

Setting	Policy	C		GD		ED		PFS		GS		IGD	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
1.a	Random	0.01	0.01	1.59	0.78	25.33	10.20	24.80	11.52	0.73	0.16	2.26	1.91
	Closest	0.01	0.01	2.14	1.26	28.37	12.29	21.98	9.49	0.74	0.19	2.53	1.71
	Rank	0.09	0.09	0.95	0.44	21.88	10.66	28.20	14.98	0.78	0.21	2.53	1.99
	NSGA-II	**0.90**	0.10	**0.04**	0.08	**16.48**	7.93	**47.52**	5.35	**0.50**	0.17	**0.26**	0.26
1.b	Random	0.01	0.02	2.38	1.21	26.20	15.20	16.80	13.29	0.75	0.13	1.92	0.96
	Closest	0.00	0.01	3.32	1.74	31.28	18.95	14.82	9.84	**0.73**	0.08	2.18	0.95
	Rank	0.06	0.01	1.68	1.39	22.03	23.91	19.44	6.79	0.83	0.17	2.13	2.12
	NSGA-II	**0.93**	0.14	**0.02**	0.05	**14.01**	9.38	**52.84**	9.46	1.04	0.54	**0.11**	0.20
1.c	Random	0.00	0.00	2.78	2.34	29.47	26.32	9.92	4.89	**0.81**	0.13	1.86	1.81
	Closest	0.00	0.00	3.11	3.55	28.58	26.45	11.08	7.49	0.86	0.16	2.10	2.17
	Rank	0.01	0.01	1.40	1.39	25.23	23.91	14.24	6.79	0.94	0.17	1.92	2.12
	NSGA-II	**0.99**	0.01	**0.00**	0.00	**16.76**	14.06	**64.34**	9.60	1.37	0.45	**0.02**	0.06

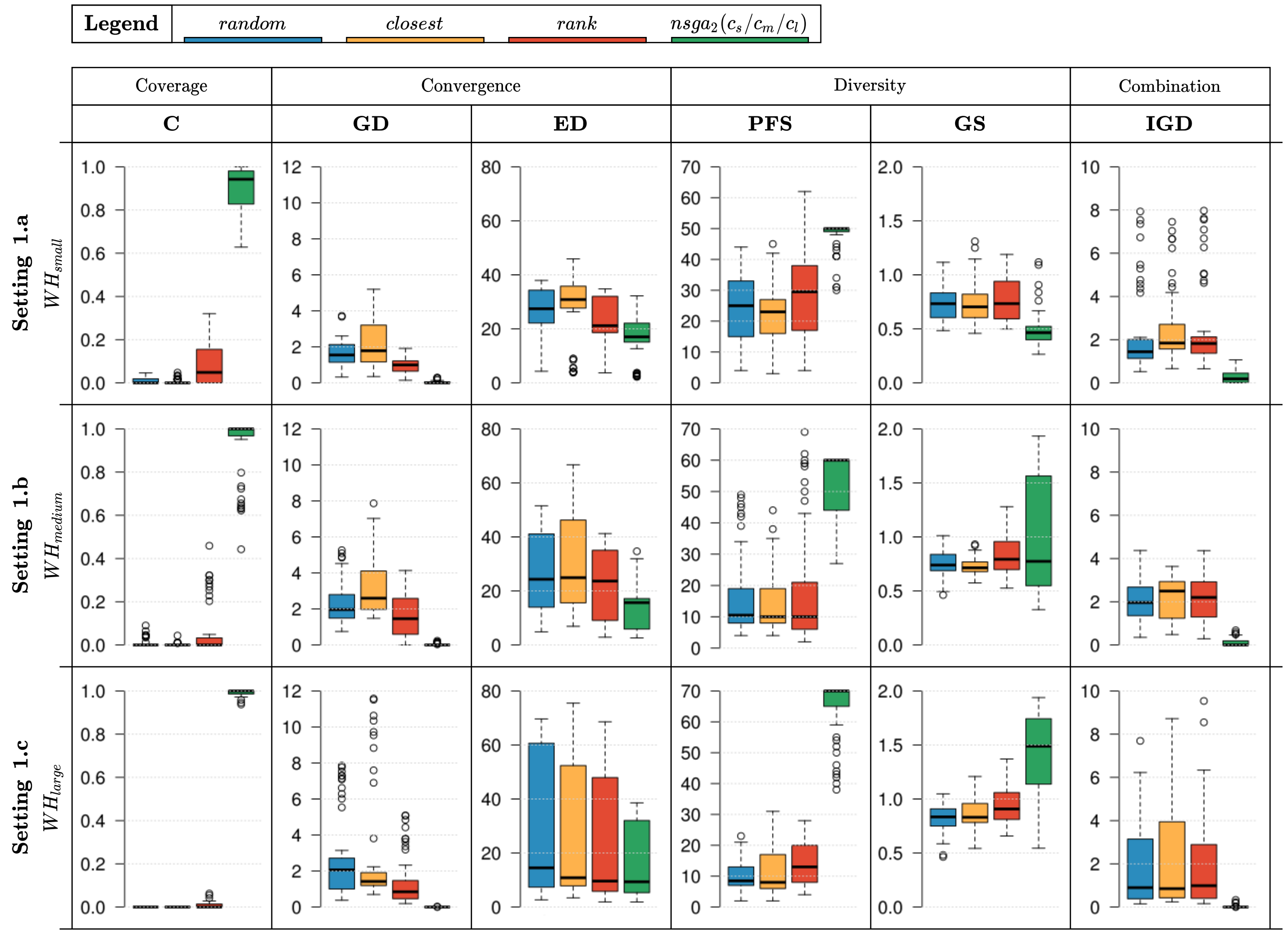


Figure 12.2: Box plots of the performance indicators achieved by the storage assignment strategies in Setting 1.a, 1.b, and 1.c.

12.5 Evaluation of the Order Picking Approach

We evaluate both versions of our ACO algorithm against the modified S-Shape heuristic and apply all approaches on the three warehouse sizes (Settings 2.a, 2.b, 2.c). Further, we randomly generate five customer orders as explained earlier and repeat the execution of the ACO algorithms ten times to reduce random effects and present mean and standard deviation values. Then, we use all generated solutions of the algorithms and strategies to calculate the reference Pareto front that integrates the best known solutions as this is required for the performance indicators. Table 12.2 summarizes the mean and standard deviation values and Figure 12.3 shows the box plots for this evaluation.

Similar to the evaluation of the storage assignment task, we first analyze the small warehouse results listed in the first row of Table 12.2 and at the top of Figure 12.3. For the small warehouse, the Pareto fronts of the ACO_3 and ACO_4 variants cover 73% and 74% of the reference Pareto front while the S-Shape strategy fails to cover even a single solution as indicated by the C performance indicator. Similar to the previous evaluation, this shows the poor performance of the alternative strategy, which could lead to a discussion about the usefulness of this selection. However, we have deliberately chosen this alternative because it is commonly used in real application scenarios and in many cases no real optimization takes place. Both ACO algorithms achieve nearly the same GD and ED values of 1.40 and 1.59, as well as 32.03 and 32.07, respectively, and the close to zero GD values show that many solutions are part of the reference front while the S-Shape strategy results in a mean GD value of 22.15. The ACO algorithms find around ten solutions per problem instance, while the S-Shape finds only three solutions per problem instance with regards to the PFS performance indicator. The S-Shape achieves the lowest, hence, the best mean GS value of around 0.84 compared to values of 0.87 and 0.99 for the ACO_4 and ACO_3 algorithms, respectively. However, the boxes of all three assessed algorithms and strategies heavily overlap and it is not statistically significant to compare the S-Shape result to the ACO ones as it contains only three solutions that are considerably worse than solutions of both ACO algorithms. The ACO algorithms achieve low IGD values of 1.91 and 2.74 for the ACO_4 and ACO_3 algorithms, respectively, indicating that both Pareto fronts converge well towards the reference front and provide diverse solutions. In summary, both ACO algorithms perform similar well in the small warehouse and we consider them better performing compared to the S-Shape strategy as they clearly outperform it in five of performance indicators.

Next, we analyze the evaluation results for all three approaches for the medium warehouse that is listed in the second row in Table 12.2 and at the center of Figure 12.3. The results for the medium warehouse show similar behavior as in the previous setting. Like in the previous setting, the S-Shape Pareto front fails to cover even a single solution in the reference Pareto front indicated by the C. The Pareto front of the ACO_3 covers approximately 69% of the reference front, while ACO_4 covers only 33%. Thus, the ACO_3 variant tends to find better pick routes than the ACO_4 variant. The Pareto front of the ACO_3 achieves the lowest GD and ED values of 1.97 and 50.11 among all computed Pareto fronts that show values of 4.30 and 54.59 for the ACO_4 variant and 31.20 and 117.42 for the S-Shape strategy. Thus, ACO_3 converges best towards the reference front, which is not surprising as ACO_3 covers most of the solutions in the reference Pareto front. The GD and ED values of ACO_4 are slightly larger than the ones of ACO_3, indicating that ACO_4 does not converge as well as ACO_3 towards the reference front. Compared to the previous setting, the GD and ED values of the S-Shape strategy increased, which may be due to the larger search space. Concerning the PFS metric, the ACO_3 variant finds about 12 solutions per problem instance, followed closely by the ACO_4 variant that finds around 11 solutions per problem instance, while the S-Shape heuristic finds only about 4 solutions per problem instance. Regarding the GS metric, the solutions of ACO_4 are slightly better distributed than the solutions of ACO_3 with values of 0.66 compared to 0.81 for the ACO_4 and 0.72 for the S-Shape strategy. With respect to the IGD metric, the ACO_3 achieves the lowest IGD values of 3.00 compared to a value of 4.15 for the ACO_4 and even 18.78 for the S-Shape strategy, indicating that ACO_3 converges well towards the reference front and offers a high diversity of solutions. In summary, again both ACO algorithms perform similarly well and outperform the S-Shape heuristic regarding all performance indicators while the ACO_3 shows a slight advantage compared to the ACO_4.

Finally, we analyze the results for the large warehouse listed in the third row of Table 12.2 and at the bottom of Figure 12.3. In the large warehouse, the S-Shape strategy is again unable to cover any solution in the reference Pareto front. The ACO_3 covers 84% of the reference Pareto front, while ACO_4 covers only 16%. Thus, most of the solutions found by the ACO_4 variant are dominated by the solutions found by the ACO_3 variant. Accordingly, the ACO_4 variant has problems to compete with the ACO_3 variant in larger warehouses. Compared to the previous setting, the GD and ED values of ACO_4 further increased to 9.78 compared to 1.56 for the ACO_3, indicating that the distances between the solutions of ACO_4 and the solutions of ACO_3 became larger. The Pareto front from ACO_3 converges best towards the reference Pareto front, as it achieves the lowest GD and ED values of 1.56 and 56.75 compared to 9.78 and 70.02 for the ACO_4 and even 41.41 and 121.35 for the S-Shape strategy. Regarding the

PFS metric, both ACO variants find about 10 solutions per problem instance while the S-Shape strategy can ony find two solutions. The GS metric indicates that the solutions of ACO_4 are marginally better distributed than the solutions of ACO_3 with values of 0.68 compared to 0.74. Finally, the Pareto front of the ACO_3 achieves the best IGD values of 5.64, signalizing that the ACO_3 converges best towards the reference Pareto front and offers diverse solutions.

In summary, the ACO algorithms outperform the S-Shape heuristic in all warehouse sizes, while ACO_3 and ACO_4 show similar performance in smaller warehouses. With increasing warehouse size, the solutions found by the ACO_3 variant dominate more and more solutions of ACO_4 variant. Hence, the ACO_3 variant starts to find better pick routes than the ACO_4 variant, while the ACO_4 variant produces slightly better distributed solutions.

In addition to the performance evaluation, we also measure the mean execution time of the approaches for solving 50 problem instances in each warehouse size. The S-Shape heuristic takes around 0.15 seconds to compute routes. The ACO_3 and the ACO_4 variant achieve fairly the same execution times in all warehouse sizes of around 1/3/6 seconds for $WH_{small}/WH_{medium}/WH_{large}$, respectively. As the warehouse size increases, the graph consists of more markets causing more ants to be deployed in each iteration. Still, we consider the ACO execution times acceptable, as they require only a few seconds to find noticeably better pick routes.

12.6 Evaluation of the Interaction between Storage Assignment and Order Picking Approaches

In the previous section, we have shown the applicability of our algorithms for storage assignment and order picking in dedicated analyses. The results indicate that both algorithms outperform commonly used strategies used in real applications for those tasks. In this section, we evaluate the interaction between our proposed algorithms by assessing them in three settings: In Section 12.6.1 we determine whether the ACO_3 performs better on the NSGA-II planned warehouse compared to the random warehouse for warehouse sizes small, medium, and large, which we refer to Setting 3.a, 3.b, and 3.c, respectively; in Section 12.6.2 we perform a similar assessment for the ACO_4 as Settings 4.a, 4.b, and 4.c, respectively; in Section 12.6.3 we evaluate whether the ACO_3 or the ACO_4 perform better on the NSGA-II planned warehouse on all warehouse sizes, which we refer to as Setting 5.a, 5.b, and 5.c, respectively. In Table 12.3, we provide a summary of all results using mean and standard deviations for all three settings. Additionally, we provide a box plot for each setting.

Table 12.2: Mean and standard deviation values of the six quality indicators Coverage (C), Generational Distance (GD), Euclidean Distance (ED), Pareto Front Size (PFS), Generational Distance (GD), and Inverted Generational Distance (IGD) achieved by the order picking strategies in Setting 2.a, 2.b, and 2.c (best mean values are shown in bold).

Setting	Policy	C		GD		ED		PFS		GS		IGD	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
	sShape	0.00	0.00	22.15	14.20	80.34	28.26	2.80	1.60	**0.84**	0.16	18.28	6.55
2.a	ACO_3	0.73	0.13	**1.40**	1.91	32.03	16.21	**10.66**	5.63	0.99	0.36	2.74	2.96
	ACO_4	**0.74**	0.18	1.59	2.64	**32.07**	16.14	9.54	3.97	0.87	0.35	**1.91**	2.52
	sShape	0.00	0.00	31.20	6.92	117.42	34.51	3.60	1.02	0.72	0.12	18.78	5.03
2.b	ACO_3	**0.69**	0.15	**1.97**	2.08	**50.11**	15.74	**12.14**	3.80	0.81	0.20	**3.00**	2.67
	ACO_4	0.33	0.16	4.30	3.62	54.59	14.43	11.30	3.23	**0.66**	0.17	4.15	2.68
	sShape	0.00	0.00	41.41	16.42	121.35	35.71	2.00	0.89	0.88	0.14	27.18	6.95
2.b	ACO_3	**0.84**	0.11	**1.56**	2.90	**56.75**	18.95	**10.50**	2.87	0.74	0.23	**5.64**	6.27
	ACO_4	0.16	0.11	9.78	5.83	70.02	21.76	10.14	3.28	**0.68**	0.21	7.28	3.91

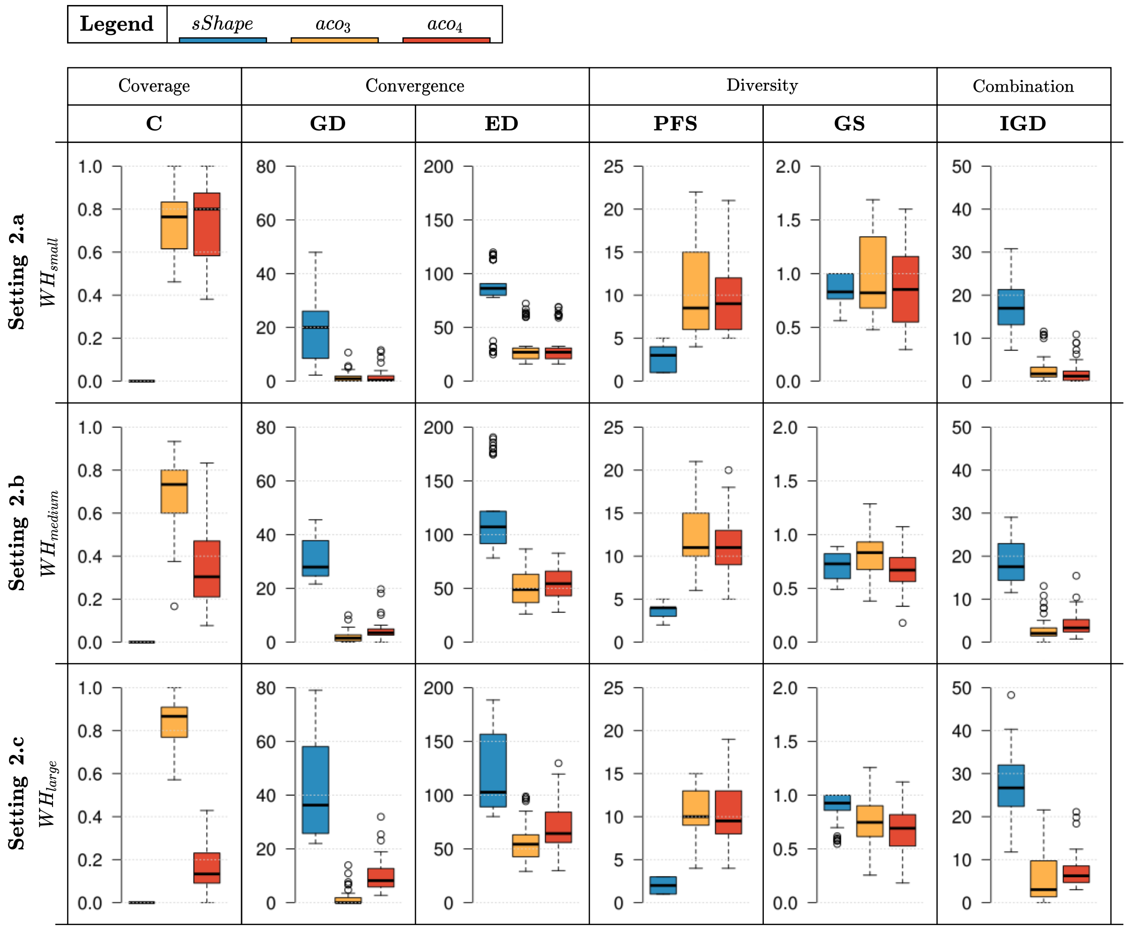


Figure 12.3: Box plots of the performance indicators achieved by the order picking strategies in Setting 2.a, 2.b, and 2.c.

12.6.1 Storage Assignment Approaches combined with ACO_3

In this setting, we apply the ACO_3 algorithm on all warehouse sizes (Setting 3.a, 3.b, 3.c) twice: once for the warehouse using the NSGA-II algorithm and once for the randomly assigned warehouse. Again, we select five random items from the product assortment and double the existing amount in the warehouse. We summarize the mean and standard deviation values for this evaluation in Table 12.3 in the first group of rows and depict box plots in Figure 12.4.

In the small warehouse, the Pareto front of the NSGA-II planned warehouse covers the entire reference front while the randomly planned warehouse does not cover a single solution in the reference front as indicated by the C performance metric. Hence, the GD and IGD values of the NSGA-II planned warehouse are down to 0 while the randomly planned warehouse produces mean values of 60.32 and 56.49 for the GD and IGD indicators, respectively. Additionally, the ED values are minimal for the NSGA-II planned warehouse with a mean value of 22.62 compared to a mean value of 215.43 for the randomly planned warehouse. The high GD and ED values of the randomly planned warehouse indicate that its Pareto front does not converge well towards the reference front. The random warehouse was able to achieve the best results only for the GS performance indicator with a mean value of 0.97 compared to 1.12 for the NSGA-II warehouse. However, as the number of solutions in its Pareto front is smaller than the ones of the NSGA-II warehouse with mean values of 9.96 and 12.20 for the random and the NSGA-II warehouse, we consider this difference as not decisive. Thus, the solutions found in the random warehouse are considerably worse than the solutions found in the NSGA-II warehouse.

In the medium warehouse, the Pareto fronts of random and NSGA-II planned warehouses achieve fairly the same performance indicator values as in the previous setting. However, the GD and ED metrics indicate that the results for the random warehouse unexpectedly converge better towards the reference front than in Setting 3.a. This could be due to the limited amount of executed problem instances and needs to be further assessed with a higher number of problem instances. Nevertheless, the Pareto front of the random warehouse is still far from converging towards the reference front.

In the large warehouse, the same observations can be made as in the previous settings, underlining that the ACO_3 variant finds better pick routes in the NSGA-II warehouse than in the random warehouse. In summary, the evaluation results show that the NSGA-II algorithm and the ACO_3 variant interact well together and the ACO_3 variant profits from the NSGA-II algorithm that ensures our four economic constraints.

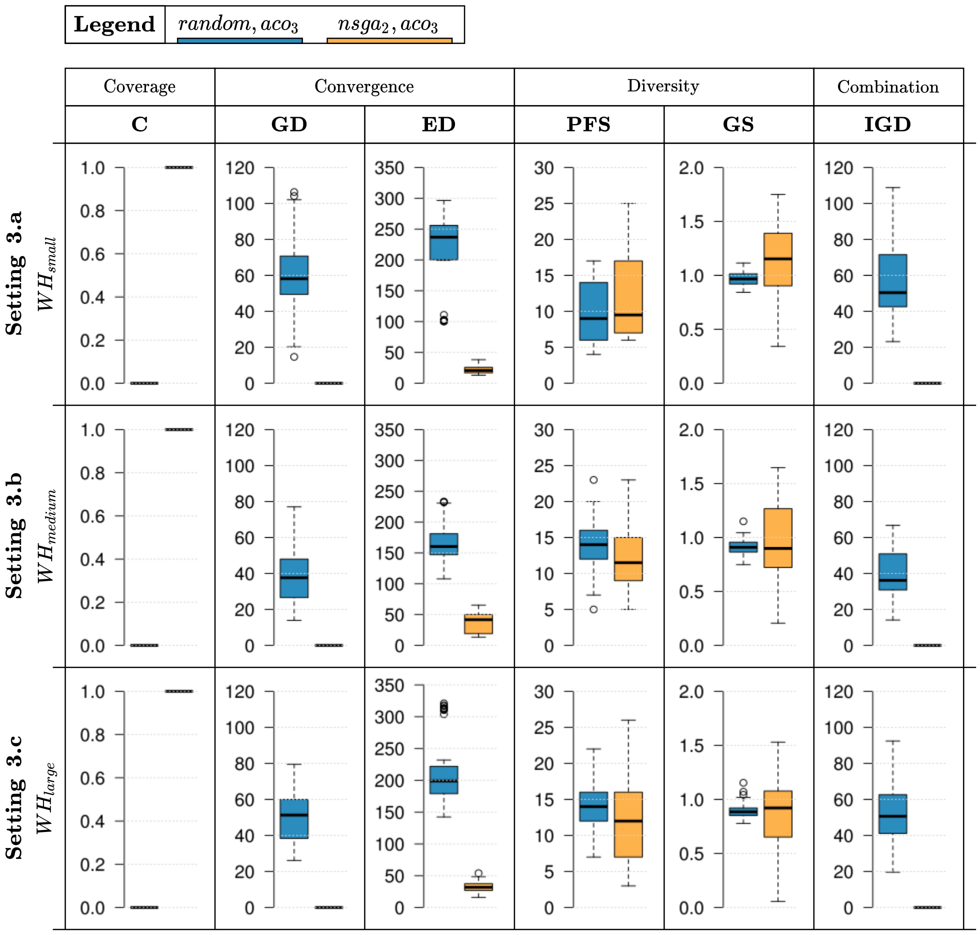


Figure 12.4: Box plots of the performance indicators achieved by the interaction evaluation for the ACO$_3$ algorithm in Setting 3.a, 3.b, and 3.c.

12.6.2 Storage Assignment Approaches combined with ACO$_4$

This setting repeats the Settings 3.a to 3.c for the ACO$_4$ algorithm. Again, we summarize the mean and standard deviation values for this evaluation in Table 12.3 in the second group of rows. Further, Figure 12.5 depicts the box plots for these three settings.

In the small warehouse, the Pareto front of the NSGA-II warehouse covers the entire reference Pareto front and the random warehouse fails to cover a single solution as indicated by the C performance indicator. Thus, all solutions found in the random warehouse are dominated by the solutions of the NSGA-II

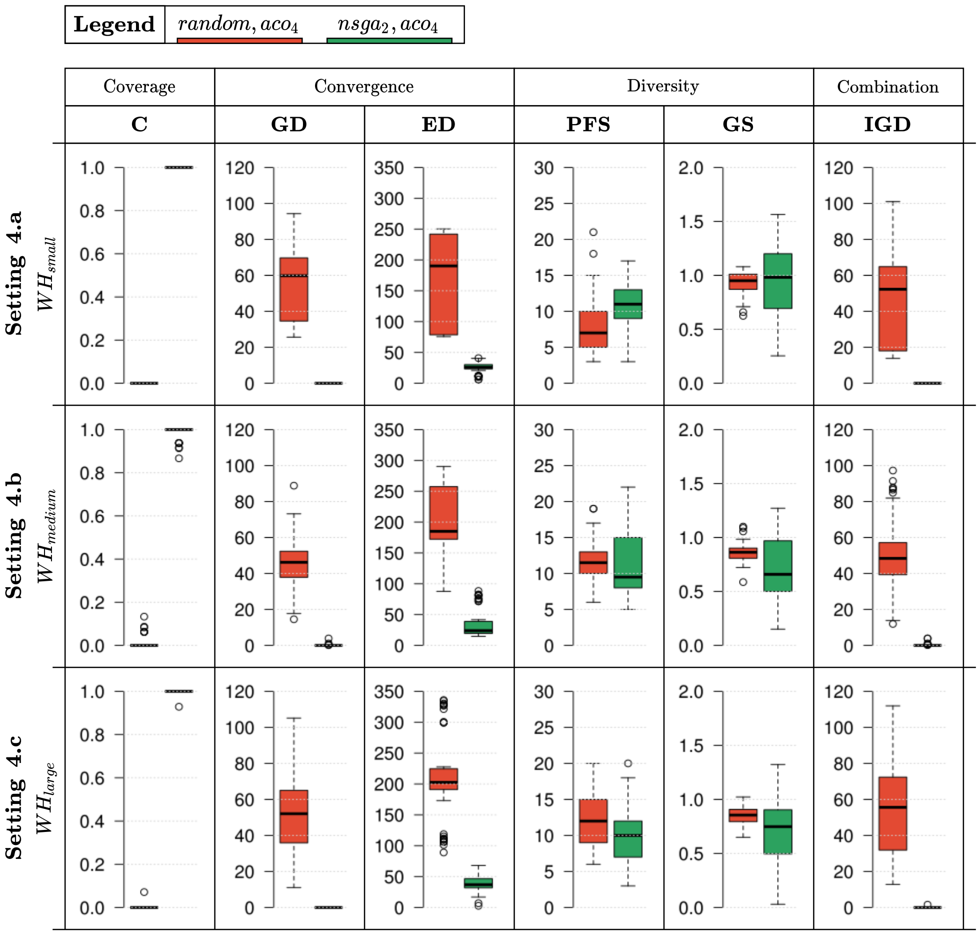


Figure 12.5: Box plots of the performance indicators achieved by the interaction evaluation for the ACO_4 algorithm in Setting 4.a, 4.b, and 4.c.

warehouse. The GD and ED values for the random warehouse are higher than the ones of the NSGA-II warehouse with a value of 54.03 and 165.75 compared to 0.00 and 25.08. This shows that the Pareto front of the random warehouse is further away from the reference front. Similar to the previous setting using the ACO_3 algorithm, the NSGA-II planned warehouse produces more solutions in the Pareto front with a value of 11 than the randomly planned warehouse with a value of 8, while the randomly planned warehouse shows a slightly smaller mean GS value of 0.93 compared to 0.96. Again, we consider this difference not decisive and conclude that the solutions of the random warehouse are noticeably worse than the solutions found in the NSGA-II warehouse.

In the medium warehouse, the Pareto front of the NSGA-II warehouse does not always cover the entire reference front with a mean C value of 0.99, while the random warehouse covers at least one solution in the reference front in 7 of 50 repetitions with a mean C value of 0.01. Thus, the ACO_4 variant occasionally finds a few solutions in the random warehouse that are comparable with the solutions found in the NSGA-II warehouse. Despite these few outliers, the results show a similar behavior as when using the small warehouse.

Similar to both previous settings, the evaluation in the large warehouse show comparable results. The NSGA-II warehouse covers all solutions in the reference front in 49 of 50 repetitions and the random warehouse manages to cover at least one solution in the reference front. In summary, the results show that the ACO_4 variant also finds better pick routes if the warehouse applies the NSGA-II storage strategy and both algorithms interact well with each other.

12.6.3 Order Picking Approaches combined with NSGA-II

This section investigates which ACO variant performs better if the warehouse applies the NSGA-II storage strategy, hence, it analyzes how well both of our proposed approaches interact with each other. Again, we apply both variants of the ACO on five randomly generated customer orders on all warehouse sizes (Settings 5.a, 5.b, 5.c). We summarize the mean and standard deviation values for this evaluation in Table 12.3 in the last group of rows. Further, Figure 12.6 depicts the box plots for these three settings.

In the small warehouse, the Pareto front of the ACO_3 algorithm covers approximately 80% of the reference front, while ACO_4 covers only 66% as indicated by the C metric. Both ACO variants converge well towards the reference front as indicated by the low GD and ED values of around 1.43 and 25.30 for the ACO_3 and around 1.52 and 26.58 for the ACO_4. Both variants find approximately ten solutions per problem instance indicated by PFS. The IGD values of both fronts are similar with values of 1.38 for ACO_3 and 0.98 for ACO_4 compared to values above 3 for the larger warehouse sizes. However, the GS values indicate that the ACO_4 variant produces a better distribution than the ACO_3 with mean values of 0.88 compared to 0.99.

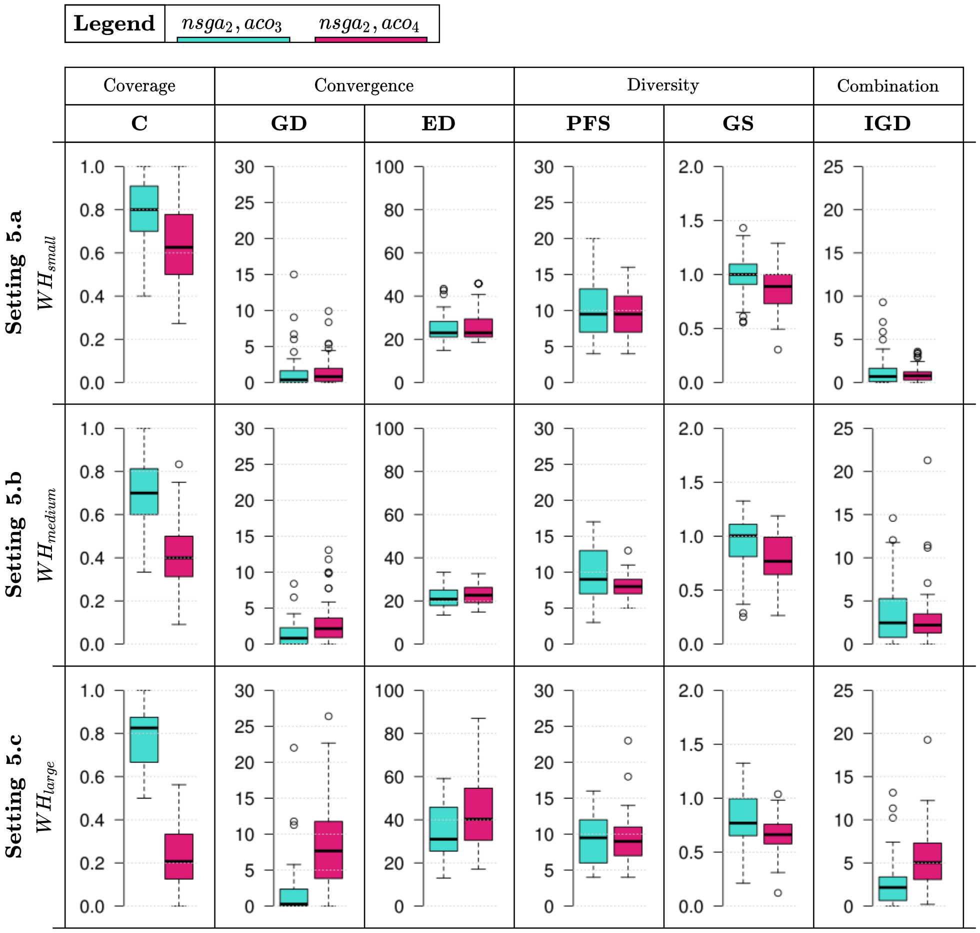


Figure 12.6: Box plots of the performance indicators achieved by the evaluation between ACO$_3$ and ACO$_4$ with the NSGA-II algorithm in Setting 5.a, 5.b, and 5.c.

Table 12.3: Mean and standard deviation values of the six quality indicators Coverage (C), Generational Distance (GD), Euclidean Distance (ED), Pareto Front Size (PFS), Generational Distance (GD), and Inverted Generational Distance (IGD) achieved by the evaluation runs addressing the interaction of storage assignment and order picking tasks in Setting 3.a-3.c, 4.a-4.c, and 5.a-5.c (best mean values are shown in bold).

Setting	Policy	C		GD		ED		PFS		GS		IGD	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
3.a	Random, ACO_3	0.00	0.00	60.32	22.94	215.43	62.38	9.96	4.10	**0.97**	0.06	56.49	22.96
	NSGA-II, ACO_3	**1.00**	0.00	**0.00**	0.00	**22.62**	7.51	**12.20**	6.17	1.12	0.35	**0.00**	0.00
3.b	Random, ACO_3	0.00	0.00	38.37	13.66	168.05	35.09	**13.76**	3.88	**0.91**	0.07	38.85	14.41
	NSGA-II, ACO_3	**1.00**	0.00	**0.00**	0.00	**36.65**	16.50	12.12	4.18	0.96	0.36	**0.00**	0.00
3.c	Random, ACO_3	0.00	0.00	50.06	14.03	213.69	54.74	**13.96**	3.56	0.90	0.07	51.35	17.53
	NSGA-II, ACO_3	**1.00**	0.00	**0.00**	0.00	**32.78**	8.19	12.02	5.40	**0.87**	0.32	**0.00**	0.00
4.a	Random, ACO_4	0.00	0.00	54.03	18.52	165.75	73.60	8.14	3.69	**0.93**	0.10	46.16	26.01
	NSGA-II, ACO_4	**1.00**	0.00	**0.00**	0.00	**25.08**	8.71	**10.26**	3.65	0.96	0.31	**0.00**	0.00
4.b	Random, ACO_4	0.01	0.03	44.61	15.52	198.83	62.11	**11.82**	2.96	0.86	0.09	48.94	22.17
	NSGA-II, ACO_4	**0.99**	0.03	**0.12**	0.57	**35.65**	22.53	11.10	4.20	**0.71**	0.28	**0.21**	0.77
4.c	Random, ACO_4	0.00	0.01	51.19	21.77	207.95	68.40	**12.04**	3.55	0.85	0.08	52.91	26.20
	NSGA-II, ACO_4	**1.00**	0.01	**0.00**	0.00	**38.08**	12.91	10.04	3.82	**0.71**	0.28	**0.03**	0.22
5.a	NSGA-II, ACO_3	**0.80**	0.15	**1.43**	2.70	**25.30**	6.61	**10.26**	4.74	0.99	0.20	1.38	1.93
	NSGA-II, ACO_4	0.66	0.20	1.52	2.11	26.58	7.98	9.32	3.72	**0.88**	0.20	**0.98**	0.93
5.b	NSGA-II, ACO_3	**0.69**	0.16	**1.40**	1.74	**21.37**	4.58	**9.84**	3.43	0.94	0.25	3.53	3.46
	NSGA-II, ACO_4	0.41	0.16	3.83	5.59	22.85	4.58	8.14	1.90	**0.79**	0.22	**3.13**	3.48
5.c	NSGA-II, ACO_3	**0.79**	0.14	**1.92**	3.81	**33.80**	12.68	9.16	3.28	0.81	0.25	**3.11**	4.40
	NSGA-II, ACO_4	0.22	0.14	8.69	6.32	43.25	17.01	**9.18**	3.54	**0.66**	0.18	5.71	3.54

In the medium warehouse, the Pareto front of ACO_3 covers approximately 69% of the reference front, while the one from ACO_4 covers only 41% indicated by the C metric, and thus, the solutions found by ACO_4 tend to be dominated by the ones from ACO_3. The GD and ED metric indicate that the ACO_3 front converges better towards the reference front with mean values of 1.40 and 21.37 for the ACO_3 and 3.83 and 22.85 for the ACO_4 variant, respectively. The GS values indicate that solutions of the ACO_4 Pareto front are better distributed with a value of 0.79 compared to 0.94 for the ACO_3.

In the large warehouse, the ACO_3 dominates ACO_4 even more with regards to the C metric showing mean values of 0.97 compared to 0.22. Furthermore, the GD and ED values of ACO_4 increased to 8.69 and 43.25, respectively, indicating that the distance between the solutions in ACO_4 Pareto front and the solutions in the reference front become larger. Again, the solutions in the ACO_4 Pareto front have a slightly better distribution than the solutions in the ACO_3 Pareto front as indicated by the values for the GS metric with a value of 0.66 compared to 0.81. However, this time, ACO_3 achieves better IGD values of 3.11 compared to 5.71, as the ACO_3 covers large parts of the reference front. In summary, we can state that with increasing warehouse size, the ACO_3 variant finds better pick routes than the ACO_4 variant while both variants find approximately the same number of solutions per problem instance. However, the solutions found by the ACO_4 are slightly better distributed than the ones from the ACO_3.

12.7 Threats to Validity

We identified the following threats to validity of our evaluation. First, the NSGA-II and ACO algorithms are evaluated in three custom build mezzanine warehouses of different sizes. However, real-world mezzanine warehouses may consist of more floors, blocks, pick aisles, and racks than specified in the warehouses used for evaluation. Nevertheless, we are convinced that our defined warehouses form a representative set for mezzanine warehouses and can easily be extended for further evaluation runs. Second, since the algorithms and strategies are evaluated in warehouses that apply either the random or the NSGA-II storage strategy, the evaluation results may not be transferable to warehouses that apply different storage strategies. Even though the product assortment, the product correlations, the customer orders, and the storage allocations are randomly generated reflecting specific characteristics of real-world mezzanine warehouses, the proposed algorithms are easily transferable to real application data. Third, we decided to compare the ACO algorithm only with one order picking strategy. This decision was made in awareness of the limited expressiveness of our results but was necessary as the majority of policies in the literature violate assumptions made for our approach. Further, we evaluate our NSGA-II and ACO algorithms only against heuristic strategies. Hence, they should additionally be evaluated against further optimization methods like other evolutionary optimization algorithms or graph-based optimization techniques. However, we decided to postpone these evaluations to future work. Finally, the limited amount of repetitions per setting do not allow for meaningful statistical tests which limits the expressiveness of our results. Still, we presented mean and standard deviation values as well as boxplots using ten repetitions. Further, we evaluated the approaches using a diverse set of metrics assessing the performance of Pareto fronts and the presented approaches appear to dominate the other approaches clearly. The combination of the repetitions and the diverse set of metrics provides a meaningful first insight into the performance of the presented approaches. However, additional evaluation runs including a larger amount of repetitions can be performed in the future to examine the results shown for their statistical significance.

12.8 Summary

In this chapter, we evaluated our second contribution, which focuses on **Goal B**: *Improving the quality of optimization strategies in complex systems-of-systems with a special attention to the field of logistics*. We answer **RQ B.2** and its subordinate question **RQ B.2.3** by analyzing the proposed approaches for the mezzanine warehouse use case. The evaluation showed that our proposed optimization algorithms integrate all real-world constraints. Furthermore, the results show that considering the interrelatedness and the direct integration of this characteristic into the optimization approaches improves the solution quality. Hence, the results have tremendous practical benefits and show that the proposed approaches take into account the complex systems-of-systems structure.

Part IV

Conclusions

Chapter 13

Related Work

In this chapter, we present the current state-of-the-art literature related to this work. Therefore, we divide the discussion of related work into four parts. First, we present approaches in the literature related to the self-aware optimization framework in 13.1. In this section, we review the literature on the main functions of the framework such as situation detection, strategy selection, and parameter optimization. We also present related approaches to fairness aspects in adaptation planning strategies and approaches to managing uncertainty during a planning phase. We then present related work on the systems-of-systems integration in Section 13.2 and discuss how the proposed approaches for the rVRP and mezzanine warehouses relate to this research are. Then, we analyze the related literature for the rVRP in Section 13.3. Finally, we summarizes the state-of-the-art literature on the optimization of mezzanine warehouses in Section 13.4.

13.1 Self-aware Optimization Framework

This section discusses related approaches on situation-awareness, meta-self-awareness, algorithm selection, and meta-optimization. The content is based on our publications [LNH$^+$21, Les20].

A recent study by Calinescu et al. [CMPPW20] has shown that situation-awareness is the main driver for the development of self-adaptive systems and is therefore still an important research topic with many open research challenges. Endsley [End17] presents a theoretical model of situation-awareness in relation to dynamic human decision making, building on research on naturalistic decision making. Fredericks et al. [FGKV19] present an approach that uses clustering to determine the current situation. They use this information for optimization techniques to discover the optimal configuration for black-box systems. Liu et al. [LKPA15] propose an approach to situation-awareness in autonomous driving that aims to improve the decision-making process in an urban environment. Rockl et al. [RRFS07] propose an architecture for driver assistance systems that uses increased environmental information to detect hazardous situations. Hardes et al. [HS19b] address communication problems in urban platooning scenarios by using the concept of situation-awareness. Porter et al. [PRF16] propose a software framework that learns optimal system assemblies in emergent software systems. Kang et al. [KCP20] analyze which history length and sensor range provide the best results for long-term situational awareness.

According to Lewis et al. [LBL+17, p.52], meta-self-awareness "leads to the ability to model and reason about changing trade-offs during the system's lifetime". Cox et al. [Cox05] research on meta-cognition, which bridges psychology and computer science. Agarwal et al. [AME+09] provide an approach that allows computer systems to reason about their own knowledge. Perrouin et al. [PMC+12] propose a rule-based approach to meta-self-awareness. They use layered MAPE-K Control Loop to optimize adaptation decisions and make an adaptive system "resilient to a larger number of unexpected situations" [PMC+12, p.1354]. Gerostathopoulos et al. [GBH+17] propose the concept of meta-adaption for cyber-physical systems, which improves the adaptation of a cyber-physical system by generating new self-adaptation strategies at runtime. Kinneer et al. [KCW+18] propose the idea of re-using knowledge from previous plans for optimization. They use a white-box approach with knowledge about the system combined with a genetic algorithm to respond to unexpected adaptation scenarios.

Kate Smith-Miles considers algorithm selection as learning problem [SM09]. She reviews the interdisciplinary literature dealing with algorithm selection and presents the developments in this research area. Kerschke et al. provide a survey on automated algorithm selection [KHNT19]. The survey covers early and recent work in this area and discusses promising application areas. Further, it includes an overview on related areas such as algorithm configuration and scheduling. Pascal Kerschke and Heike Trautmann contribute an approach for automatic model construction for algorithm selection in continuous black-box optimization problems [KT19]. The goal of this approach is to reduce the required resources of the selected optimization algorithms. Kotthoff et al. apply algorithm selection on the TSP problem [KKHT15]. They apply two existing TSP solvers and show that they perform complementary in different instances. The authors design algorithm selectors based on existing TSP features from the literature as well as new features. Bischl et al. propose a benchmark library for algorithm selection [BKK+16]. They define a standardized format for representing algorithm selection scenarios. Further, they provide a repository containing data sets from the literature to compare proposed approaches.

Neumüller et al. [NSW+12] present an implementation of parameter meta-optimization for the heuristic optimization environment *HeuristicLab Hive*. Their approach minimizes the expert knowledge required to adapt the parameters of a meta-heuristic. In their evaluation, Neumüller et al. showed that the obtained parameter combinations in some cases deviate strongly from the usual settings. However, their approach mainly covers single-objective optimization, whereas a multi-objective problem can only be assessed using a normalized and weighted sum of objectives. Feurer et al. [FSH15] improve the Sequential Model-based Bayesian Optimization (SMBO) used for tuning the parameters of machine learning algorithms involving meta-learning. Using the knowledge from past optimization runs, they showed significant improvement in the SMBO algorithm. Zhang et al. [ZHO+18] address the problem of release planning, which means the process of deciding which features to integrate into the next version of a software release. The authors perform a study on various meta- and hyper-heuristics used for multi-objective release planning. They use different hyper-heuristic algorithms to decide on search operators for meta-heuristics to improve solution quality and compare their performance. Chis et al. [CVV13] use the Framework for Automatic Design Space Exploration (FADSE) to compare the performance of different multi-objective meta-heuristics. The authors show that all algorithms find similar Pareto front approximations with good solution quality. Similarly, Vinctan et al. [VCIC15] deal with design space exploration by implementing a meta-optimization layer for the tool FADSE. With this approach, it is possible to introduce a meta-optimization function that can use multiple meta-heuristics simultaneously by switching between them at simulation runtime. In the evaluation, the authors show that their meta-optimization approach leads to better results than running two different meta-heuristics independently and combining their results.

Another research direction related to this work is the area of Auto-ML. As the name suggests, automated machine learning focuses on automating machine learning mechanisms by using pipelines in combination with hyperparameter optimization to reduce manual effort. Reinbo, for example, is an Auto-ML framework that uses task pipelines and implements reinforcement learning and Bayesian optimization to automatically determine the parameters [SLB19]. A similar approach is used by Chai et al. who propose an Auto-ML framework that covers the common problem of data drift in machine learning [CCZL19]. Thornton et al. propose a mechanism for hyper-parameters selection and optimization in the context of classification algorithms [THHLB13]. Finally, Li et al. address the problem of tuning hyper-parameters using random search combined with adaptive resource allocation and early-stopping [LJD+17].

Finally, the research on fitness landscape analysis is also related to this work. The idea of this research area is to gain a better understanding of the search space of optimization problems to derive problem-specific information and tune the selected algorithm on the characteristics of the problem [PA12]. A special focus of this research is on heuristic solution approaches. However, this analyses require much computation time, which could also be used to compute an actual solution for the problem to be solved. In contrast to the static fitness landscape analysis, the dynamic landscape analysis additionally considers the performance of the desired algorithm on a specific problem and aim on recommendations on the most promising algorithm [WLZY18]. The concept of fitness landscape analysis is applied on diverse problem statements such as quadratic assignment problem [MF00], multi-dimensional Knapsack problem [TPC08], traveling thief problem [YMK+18]. Further, this concept is used to improve the performance of neural networks [NFP21] or to further enhance the approaches on Auto-ML [PdSOP20]. Finally, Sun et al. perform a study on the optimal selection of fitness landscape analysis metrics for continuous optimization problems [SHKM14].

This work delineates from the presented related work as follows: All mentioned approaches already cover parts of our proposed framework, such as a rule-based meta-self-aware approach, situation-awareness, determining the optimal configuration of a system, analyzing the search space of the problem or performance comparison of optimization techniques. However, there is no other work that integrates all these aspects into one framework. The combination of a multi-layered framework with the LRA-M Loop and the integration of adaptation planning strategies, situation-awareness, strategy selection, learning approaches, and optimization techniques make the proposed approach unique and a valuable contribution to the research community. The fitness landscape analysis, presented as last aspect in this section, is currently not integrated in this work, but provides a promising possibility to enhance the framework and kick-start the optimization process.

Fairness in Adaptation Planning Strategies

This section discusses related work in the are of compensation models to ensure fairness, particularly in the area of ITS containing platooning compensation models and vehicle sequence optimization within a platoon. Furthermore, this section presents approaches to compensation-based system integration from the SISSY research domain. This section is mainly taken from our publication [LKS+21]

Besides the technological aspects of platooning, the SARTRE project [RCC10] also included studies on incentives and compensation models. In the *monthly subscription* model, that is, a market compensation model, customers pay a monthly fee, which also compensates the lead driver for his effort. Similarly, in the *pay-as-you-go* model, users pay the platoon leader a fee to join the platoon over a predefined distance or pay per usage. To compensate the users of platooning for the paid fees, the SARTRE researchers recommend a *Free Sponsored Benefits* model, i.e., governments offer free services—free parking or access to car pool lanes—to make platooning more appealing. The *taking turns model* assumes a large user base: Users are incentivized to act as a platoon leader as this is the only possibility to earn the right to be in an inner-platoon position in the future. Whereas the first two models are examples for direct compensation of the platoon leaders, the last two are indirect compensations for users of platoons. The TNO project [JZBdK15] differs substantially from SARTRE as it only targets trucks and as it does not allow a platoon to be longer than two vehicles. TNO considers Logistic Service Providers (LSP) and Platooning Service Providers (PSP). The *scheduled platooning* model describes the idea that LSP (forwarder, shipper, haulers) use platooning whenever two trucks of the own company at least partially travel together. The *on-the-fly platooning* model describes dynamic inter-company platooning. This requires compensation mechanisms; however, TNO does not further specify them. A third model integrates PSP as instances that coordinate platooning, handle administrative issues, and transfer compensation payments. Peloton uses a direct compensation model that offers platooning as a usage-based service [Pup16]. The cloud-based Network Operations Centre coordinates platooning and assignments of vehicles to platoons. When situated in the fuel-saving position, a vehicle pays a per-mile fee that includes compensation payments for the platoon leader.

Research on the topic of intra-platoon vehicle sequence optimization deals with finding an optimal ordering of the platooning vehicles. Depending on the exact use case, different variables can be optimized. Hao et al. [HWW$^+$17] investigated the optimal joining position for new vehicles in order to minimize acceleration, deceleration, and cruising maneuvers when opening/closing gaps, which leads to a more energy-efficient usage of platooning. To achieve this, they formulated a bi-level integer programming model. Similarly, Liang et al. [LAG11] also analyzed the effects of ordering intra-platoon vehicles in different ways. They focused on the mass of Heavy-duty Vehicle (HDV) as an important factor which influences the driving characteristics. The results of both strategies show that the sequence of vehicles inside a platoon can have a considerable impact and can be used to achieve higher fuel efficiency.

However, previous research on platoon formation neglects the aspect of evenly distributing the benefits of platooning in terms of fuel efficiency. This thesis contributes to this research by proposing six different rotation strategies that ensure fairness among the participants in a platoon, as presented in Section 7.7. We consider these strategies as adaptation planning strategies, which are part of the second layer in our proposed system model in Section 7.5. Hence, we allow the rotation strategies to be switched depending on the current situation the platoon is in, for example, the number of lanes or vehicles on a road section.

Planning under Uncertainty

This section analyzes related work on adaptation planning under uncertainty and is mainly taken from our publication [LHKK21b]. Human-in-the-loop interaction is often seen as a potential source of uncertainty [SWM19,MHAW16]. Cámara et al. discuss different types of involving humans in self-adaptive systems but focus on integrating humans for doing tasks that are difficult or infeasible to automate [CGMS17]. Here, the human is seen in the function of a system administrator. Similarly, the framework of Gil et al. provides a design of human participation in the control loops [GPFA16]. Huang and Miranda present an approach to adding users' intentions through neural input into the adaptation decision [HM15]. Consequently, this enables to adjust the system's behavior to the goals of the users. Similarly, Becker, Hähner, and Tomforde present an approach to integrate flexibility through incorporating changing user goals in a learning-based adaptation decision making [BHT12]. Cámara, Moreno, and Garlan define a modeling approach for reasoning about the humans' capability for being involved in self-adaptation. The modeling approach relies on stochastic multiplayer games [CMG15].

Another research stream focuses on the reasoning of adaptation under uncertainty. POISED [EKM11] supports reasoning on uncertainty for the adaptation decision by evaluating the consequences of uncertainty using possibility theory. Additionally, POISED integrates this assessment in the adaptation logic to include. Moreno et al. present an approach for handling the latency resulting from reactive adaptation [MCGS16] based on Markov Decision Processes (MDP) with probabilistic model checking for improving proactive adaptation by explicitly handling the uncertainty resulting from the MDP. Recently, Moreno et al. presented tactics to reduce uncertainty coming from simplified design assumptions, noise, model drift, context issues, human-in-the-loop, or decentralization [MCGK18]. Gerostathopoulos et al. present an approach to handle uncertainty resulting from noisy system outputs using Bayesian Optimization with Gaussian Processes [GPB18]. They evaluate their approach using the traffic navigation self-adaptation exemplar CrowdNav [SGPB17]. Kinneer proposes an approach for planning in unexpected situations by using prior planning knowledge based on genetic programming and reusing existing plans [KCW$^+$18]. SimCA* provides a control-theoretic approach that provides guarantees for uncertainty related to system parameters, component interactions, system requirements, and environmental uncertainty [SWM19]. A detailed discussion on how to handle uncertainty in self-adaptive systems can be found in [EM13].

Various authors deal with the topic of applying utility functions as decision criteria for self-adaptive systems [FKH11, GSCG17, KPP08, WTKD04]. Glazier et al. [GSCG17] state that each adaptive system incorporates a utility function. This could be, for example, when applying a rule-based approach, the developers have to determine an order of the applied actions and, thus, a utility function can be used to model the decision logic. According to [KPP08], a utility function provides a mathematical function which calculates a measure of goodness to determine a comparable score for possible adaptations. The scores of available options enable self-configuration of the system by selecting the highest score which results in the highest user satisfaction [GSCG17].

However, there is a limited body of work that addresses the question of how to switch between different utility functions depending on users preferences and the current situation the system needs to face. In this thesis, we apply the concept of utility functions to deal with uncertainty in ITS and propose a set of functions. We consider these utility functions as part of the second layer of our multi-layer system model proposed in Section 7.5. Thus, by applying our proposed framework, we fill this research gap and enable dynamically switching between adaptation planning strategies, that is, between utility functions.

13.2 Systems-of-Systems Integration

This section summarizes related work in the area of systems-of-systems involving their integration. Therefore, this section relates to the vertical and horizontal systems-of-systems approaches of the rVRP and mezzanine warehouses in Chapter 8 and Chapter 9. It is mainly taken from our paper [LKS$^+$21]. The focus of the SISSY research lies on systems of systems, federations of systems, or interwoven systems [THS$^+$14]. The integration of those large scale, heterogeneous entities for enabling a cooperative behavior is a very challenging task [BTW14]. We classified coordination mechanisms for this integration into decentralized approaches—selfish behavior, altruistic behavior, negotiation—

and (pseudo-)central approaches—enforcement of central decision making, rewards/incentives [LKT19a]. First, selfish behavior might lead automatically to a coordination of the instances due to interaction awareness [LCF+15] as each entity tries to optimize its benefits through coordination with others. Information dissemination can help to lower the risk of potentially conflicting decisions. Second, to overcome the issue of conflicting adaptation plans if selfish entities are not interaction-aware, mechanisms must ensure that such entities act cooperatively [LCF+15]. Still, situations can occur where agents may disagree but still need to find a consensus. Then, negotiation techniques–such as auctions [JFL+01], or bio-inspired approaches (e.g., [RKVB14])—might support the integration of entities to a shared system. Further, there are scenarios where a central or pseudo-central instance is necessary. Approaches based on leader election for choosing one specific node that acts on behalf of the group can help to enforce a central plan. Instead of forcing the resources to obey a given plan, incentives convince the resources to choose from adaptation alternatives specifying degrees of freedom.

In emergent-based approaches, the system is fully decentralized as agents act autonomously without using explicit coordination or negotiation techniques. For coordination purposes, this generally refers to simple scheduling schemes (see [Pin12] for an overview). Alternative solutions include concepts from Organic Computing (e.g., [SMM08]). However, in some situations, incentives are used as a mechanism to guarantee participation. In literature, different approaches to counter the negative effects (such as decreased willingness to participate or unfairness) are known, including compensation based on (crypto) money, trust values, scheduling priorities, or reputation. A corresponding overview can be found in [Req05, ZYS+16].

Another stream of research focuses on a fair distribution of benefits and resources. Pitt et al. [PBM14] present an approach to common-pool resource management based on Rescher's theory of distributive justice [Res02]. Voting functions collectively determine the rank order in which resources are allocated; hence, the systems self-organize the allocation method. Similar, Garbiso et al. [GDC+17] adopt the theory of distributive justice to ensure fairness in a use case of clusters of connected vehicles.

This thesis ties in with these research areas, as it aims at optimizing problems that can be considered as system-of-systems. The rVRP discussed in Chapter 8 can be considered as a vertical system-of-system, since each VRP system contains several TSP systems that need to be optimized. In this thesis, this vertical systems-of-systemsstructure is taken into account by designing a two-layered approach that allows both systems to be optimized individually. This leads to better overall results since both systems can be treated independently and the most appropriate optimization algorithm can be selected for each system. Further, the optimization of storage assignment and order picking processes in mezzanine warehouses can be considered as horizontal system-of-system, since both processes coexist and cooperate, affecting the output quality of each other. We address this interdependence by designing optimization algorithms for each of the processes with particular attention on their interaction. This unique approach and the integration of the interaction already at the design time represents the unique selling point of this work.

13.3 Optimization of the Rich Vehicle Routing Problem

This section presents an overview of the state-of-the-art of optimizing the VRP and is mainly taken from our publication [LKK+21a]. The TSP and VRP are well-known and highly researched transportation problems that were first mentioned in the last century: the TSP in 1930, and the VRP in 1959. Hence, the literature provides many different approaches to both of the problem statements. Besides the classical VRP, that assigns customer orders to vehicles and optimizes their tours, several extended VRP versions exist. These versions include additional requirements to the VRP such as capacities of vehicles, time windows, and pickup and delivery behavior. In the following, we introduce the most common variants of the VRP and provide a list of related work focusing on this specific problem statement. Cordeau et al. [CLSV07] introduce a capacity constraint for all vehicles of the fleet that must not be exceeded and thus define the C-VRP [FTV94, AHM13, CLSV07, FLL+05, LN87, QTY10, GC09, WZZL18, RSM20, SWH11, lCkYmW06, BM04, DX06, YYY09, BA03, BB03, GLP02, VCG+12, CM13]. In the VRP-TW, each customer order can be defined using additional time windows that refer to opening hours of the location which need to be met by the delivery vehicle [CLSV07, HI20, CC02, Ski11, CM12, BS03, RMLG07, FMP07, GG10, KTSA14, ENOBTMG16]. The VRP-PD provides the possibility to return goods to depots or transport them from one location to another one and to place multiple pickup and deliveries at one location [DDE+02, MJMBMD17, Aa06, WS03, CHD07, CEE16, Çat09, PDG96, TG10, SR16]. Further, the combination of time windows and pickup and delivery results in the VRP-TW-PD [LL02, WZW+16, DHR00, TYA+17, Pan05, WC12, CMMF17] All versions of the VRP are highly researched on and the literature provides a large amount of approaches to tackle these problems. We are aware, that this summary of related work is only an excerpt and does not provide a complete overview of all relevant literature in this field, but represents a spectrum of the main research streams in the area of these specific problems.

In the following, we analyze relevant literature of the last five years that explicitly covers multiple objectives as part of their VRP or TSP approach. [TGBM17] addresses multi-trip VRP with intermediate depots and time windows. The paper provides a robust Mixed-Integer Linear Programming model and addresses the following objectives: travel distances, vehicle costs, and earliness and tardiness penalty costs of services. They solve their model using CPLEX. The authors of [BTASG19] also address a multi-trip VRP in the domain of urban waste collection. They seek to minimize cost objectives, such as traversing costs, employment costs, and exit penalties from permissible time windows. Unlike the previous papers, they use Simulated Annealing to solve their problem.

[DBM$^+$20] addresses a multi-objective set orienteering problem using clusters of customers. The authors assign a predefined profit amount per visit to each customer in a cluster and specify a maximum service time. Their approach has two objectives: maximizing customer satisfaction and maximizing profit. The advancement of this method is to incorporate customer satisfaction objectives instead of standard cost-based objectives.

A multi-objective model of the capacitated VRP for perishable goods is proposed in [BDMK21]. The objectives of this model are to minimize the quality degradation of goods and to minimize the delivery costs. The authors propose an m-ring star distribution network with two types of vehicles and customers, and apply NSGA-II and Strength Pareto Evolutionary Algorithm (SPEA2) with the same time complexity statements as in the previously mentioned paper. The evaluation shows that NSGA-II performs better in terms of quality and costs when using two types for vehicles.

The authors of [MBD$^+$21] deal with a multi-objective ring tree problem with secondary sub-depots. They specify a fixed node as depot and define other primary and secondary sub-depots in combination with three types of customers. The objectives include minimizing the total routing cost and minimizing the number of type 3 customers. The authors use a discrete multi-objective ant lion optimizer and showed that their approach has better efficiency for most test instances in their evaluation.

Another set of studies focuses on green approaches to VRP variants. First, [THS$^+$18] addresses a multi-trip green capacitated arc routing problem. The authors aim to minimize the total cost, which consists of routing costs, vehicle costs, and greenhouse gas generation and emission cost. They use a hybrid GA with Simulated Annealing for generating initial solutions. The authors do not specify the time complexity of their approach, but show that their solution performs desirably within a reasonable computation time. Second, [THWM20] deal with a green VRP with intermediate depots and integrate urban traffic conditions, fuel consumption, time windows, and uncertainty in demands. They model this problem as robust Mixed-Integer Linear Programming model and solve it using CPLEX. The integration of urban traffic conditions is a particular advance of this work. Third, [ATD$^+$21] proposes a Mixed-Integer Linear Programming model for the green inventory routing problem with time windows. They attempt to minimize the total cost, which consists of fuel consumption, driver cost, inventory cost, and vehicle cost. The authors use an original and an augmented Tabu Search as well as Differential Evolution.

The last set of related works from recent years covers the integration of uncertainty in the pickup demand. First, [RMW17] addresses uncertainty in urban waste collection and models the problem as a two-stage multi-objective transportation problem. They model uncertainty as grey parameters and apply a procedure to reduce them to real numbers. They solve their model using revised multi-choice goal programming. Second, [RMWG17] address a multi-choice multi-objective transportation problem and model cost, demand, and supply as multi-choice parameters. They reduce their problem to a multi-objective transportation problem by introducing binary variables and applying revised multi-choice goal programming. Third, [RMW19] addresses a multi-objective multi-item fixed-charge solid transportation problem and incorporate fuzzy-rough variables as coefficients of their objective functions and constraints. They use a fuzzy-rough expected-value operator to transform the problem into a deterministic one, and apply weighted goal programming and fuzzy programming to find solutions. Fourth, [BTGPMK19] also addresses the urban waste collection problem with uncertainties and models the problem as a robust bi-objective multi-trip periodic capacitated arc routing problem under demand uncertainty. They integrate cost and tour length objectives and solve their problem using CPLEX and a multi-objective invasive weed optimization for real-world problem instances without defining the time complexity. The particular advance of these approaches is the general applicability of their approaches to model uncertainty.

In line with the observation of [Pig13], our analysis of related work shows that existing approaches fail to address the combination of different aspects of the rVRP in such a way that all relevant requirements of a real-world application are considered simultaneously. Moreover, related approaches neglect the vertical systems-of-systems structure of the general VRP problem and, thus, do not treat it explicitly, which provides opportunities for progress in this research direction. In this work, we focus on this research gap by integrating a variety of real-world requirements of the rVRP. In addition, we explicitly address the systems-of-systems structure of the problem by designing the two-layer approach that will enable hybrid optimization methods in the future.

13.4 Optimization of Warehouse Processes

This section presents relevant literature on optimizing storage assignment and order picking in warehouses and is based on our publication [LMK+21a]. In the literature, diverse storage assignment policies exist such as the dedicated and the random storage policy [BIH19], the closest open location storage policy [DKLDR07], rank-based storage policies [PS99]. Further, class-based, golden zone, and family grouping storage policies are introduced in the literature [DKLDR07, PSH05]. Additionally, diverse approaches apply optimization techniques. [SKG12] propose a particle swarm optimization algorithm for warehouses that deploy the class-based storage policy. [Kov11] presents a mixed integer programming model for optimizing the storage assignment problem for class-based assigned warehouses. [KBW+10] apply local search algorithms for reorganizing the products in the warehouse to keep it operating efficiently. [LCL08] propose a multi-objective genetic algorithm for optimizing the storage assignment problem in automated storage/retrieval warehouses.

Similarly, heuristic policies exist for the order picking problem such as the S-Shape, Return, Mid-Point, Largest Gap, and Combined heuristic [Pet97, RdK01, Vau99] Besides, [RR83] presents an optimal algorithm using dynamic programming to find the shortest pick route in a single-block warehouse. Additionally, [DRS98] propose a mathematical model in combination with construction heuristics and apply Tabu Search to construct order picking routes. [EÖ12] present an integer programming model for optimizing the order picking problem. [XGN+10] propose an Max-Min Ant System (MMAS) algorithm for optimizing machine travel paths in automated storage/retrieval warehouses. [CWQX13] propose an ACO algorithm that detects congestion situations that arise when multiple order pickers traverse the same pick aisle simultaneously.

Finally, related work also assess the interaction of storage assignment and order picking approaches. [PS99] and [vGRCdK18] provide an overview of well-performing combinations of storage assignment strategies and routing heuristics. [MGPR07] analyze different parameters that affect the travel time in single-block warehouses that deploy the class-based storage policy. [SAAS14] study the effects of parameters on the travel distance in multi-block warehouses.

Our work delineates from these existing approaches in several aspects. First, our work applies optimization techniques rather than relying on a policy to select suitable storage racks or shortest pick routes. Second, compared to existing optimization approaches, our work integrates multiple objectives at once, considering both economic and ergonomic constraints, while most other approaches focus on a single economic objective. Finally, unlike existing work that address the impact of storage assignment and order picking tasks, we designed algorithms that optimize the objectives of both tasks. Hence, our approach optimizes storage assignment and order picking with respect to the interdependence of both processes, while other works only provide well-performing combinations of algorithms or perform parameter tuning.

Chapter 14

Conclusion

This chapter concludes the thesis by briefly summarizing its contributions, which are structured around two research goals. First, we examine **Goal A**, which focuses on the design and implementation of a self-aware optimization framework for adaptation planning strategies. To approach this research goal, we first analyze the specific characteristics of adaptation planning strategies in the example domains ITS and logistics. Based on this knowledge, we design a self-aware optimization framework that integrates situation-awareness, algorithm selection, and parameter tuning to reduce the required expert knowledge and manual effort. This framework is designed to be generally applicable to diverse domains due to its component-based structure and the possibility to provide domain-dependent information in the DDM. In addition, we propose adaptation planning strategies that ensure fairness for platooning participants, and we design utility functions that account for uncertainty in the planning stage. We apply our contributions on prototypical use cases from the ITS domain but they can also be generalized to other domains such as smart grid. Second, we address **Goal B**, which focuses on improving the quality of optimization strategies in complex systems-of-systems, with particular attention to the field of logistics. In this contribution, we first analyze real-world use cases and define constraints and restrictions for the formulation of the problems. Then, we analyze the system-of-systems structure of the addressed use cases and identify vertical and horizontal structures. We use this knowledge to design a workflow for optimizing vertical systems-of-systems and propose to incorporate the interdependence of horizontal systems-of-systems in order to improve the output quality. We apply these contributions on the prototypical use cases of rVRP and mezzanine warehouses from the domain of logistics. However, we are convinced that the contributions can also be generalized to other domains such as intelligent computer networks. In the following, we briefly summarize the contributions related to the two main research goals and refer to the associated research questions.

Contribution 1: **Analysis of Specific Characteristics of Adaptation Planning Strategies**

As a first contribution, we analyze example use cases from ITS and logistics and derive a set of characteristics that are important for the application of adaptation planning strategies. Hence, this contribution addresses the research question **RQ A.1** of **Goal A**. We choose platooning coordination and route planning in highly dynamic environments as representative of ITS. In these case studies, we observe that the choice of adaptation planning strategies depends on the current situation and show the importance of fairness and uncertainty aspects. Further, we analyze rVRP and storage assignment and order picking in the field of logistics and identify the system-of-systems structure of complex problems.

Contribution 2: **Component-based Framework for Self-Aware Optimization of Adaptation Planning Strategies**

As our second contribution, we design a novel self-aware optimization framework for adaptation planning strategies. This contribution focuses on the research questions **RQ A.2**, **RQ A.3**, and **RQ A.6** of **Goal A**. The proposed framework is generalizable to a wide range of use cases because we placed it on top of an application and its adaptation planning strategies, and defined a DDM to apply the framework to any use case. The framework automatically identifies the current situation of the system, selects the most promising adaptation planning strategy, and tunes its input parameters. Therefore, it contains four components: (i) Coordination, (ii) Situation Detection, (iii) Strategy Selection, and (iv) Parameter Optimization. To achieve this, the framework implements concepts from the SeAC and is capable of learning and reasoning. We evaluate our proposed framework and its components on the example use case platooning coordination and compare it to state-of-the-art coordination strategies.

Contribution 3: **Fairness-Ensuring Adaptation Planning Strategies**

Our third contribution is related to research question **RQ A.4** of **Goal A**. We analyze which important characteristics of adaptation planning strategies should be handled and propose to address fairness in the used adaptation planning strategies. In this contribution, we analyze the example use case platooning and define six mechanisms for rotating the sequence of platoons to ensure fairness within a platoon. These mechanisms can be categorized as adaptation planning strategies because they change the composition of existing platoons. We analyze the mechanisms to ensure equal distribution of positive and negative effects for participants.

Contribution 4: **Addressing Uncertainty in Adaptation Planning Strategies**

In our fourth contribution, we propose to consider uncertainty directly in the adaptation planning strategies as well. This contribution addresses research question **RQ A.5** of **Goal A**. In this contribution, we propose a methodology to account for uncertainty and incorporate uncertainty measures into adaptation planning strategies. We address the example use case of fueling planning along a route and design six utility functions considering different aspects of route planning. Further, we integrate uncertainty measures for dynamic fuel prices by adding penalties for longer travel time or longer distance to the next gas station. We show the positive impact of the mechanism and the utility functions on reducing uncertainty in our use case.

Contribution 5: **Optimization of Nested Systems-of-Systems**

In our fifth contribution, we research on the optimization of nested systems-of-systems on the prototypical use case rVRP. This contribution refers to research question **RQ B.1** and its subordinate research questions of **Goal B**. We define a set of constraints and objectives that must be considered to formulate a real-world rVRP. We then analyze the nested structure of the problem and design an integrated workflow. This workflow allows both systems to be optimized individually, flexibly, and interchangeably. As optimization approaches, we apply the GA and ACO algorithms to both nested systems and compare the performance with state-of-the-art optimization algorithms for this use case.

Contribution 6: **Optimization of Coexisting Systems-of-Systems**

In our last contribution, we study coexisting systems-of-systems by optimizing typical processes in the example use case mezzanine warehouses. This contribution addresses the research question **RQ B.2** and its subordinate research questions of **Goal B**. Similar to the previous contribution, we define which constraints and objectives must be considered to formulate a real-world problem. Then, we analyze the coexisting system-of-systems structure of this problem and design an integrated workflow that optimizes all constraints and aims for a good overall system performance, taking into account the interdependence of both systems. We apply NSGA-II for storage assignment and ACO for order picking and adapt them to the specific requirements of the horizontal system-of-systems structure. Finally, we show the positive impacts of this approach, which increases the performance of both processes by integrating their dependencies in the optimization algorithms.

The presented contributions represent an important advance for both the academic research community and practical applications. To the best of our knowledge, we are the first to design a self-aware optimization framework for adaptation planning strategies. The integration of situation-awareness, algorithm selection, parameter tuning, as well as learning and reasoning into one framework is a major advancement of the state-of-the-art. The promising results of the framework, which performs close to the gold standard in our evaluation, demonstrate the importance of this contribution and offer new opportunities to address the underlying problems from a software engineering perspective. In addition, our proposed adaptation planning strategies to compensate negative effects of platooning provide a major improvement, which could lead to higher acceptance in society and more likely adoption of the technology in the real world. Our mechanism and utility functions for coping with uncertainty are an important step to improving the capabilities of SASs in an increasingly turbulent environment. Moreover, our contributions to nested and coexisting systems-of-systems optimization in logistics provide a major contribution to research and practical applications. Our proposed workflow for nested systems-of-systems and consideration of the interdependence of coexisting systems-of-systems are important contributions to the state of research in logistics as well as in the systems-of-systems domain. Finally, we select real-world use cases for our approaches and cooperated with industrial partners, highlighting the practical relevance of our work. Considering real-world use cases and reducing expert knowledge and manual effort in selecting the most appropriate strategy and its parameters is a major step in bridging the gap between academia and practice. Our two-layer approach on the rVRP is integrated into our partner's software. In conclusion, the contributions of this thesis have spawned several research projects such as a long-term industrial project on optimizing parcel delivery funded by *Bayerisches Verbundforschungsprogramm (BayVFP) – Digitalisierung* and further collaborations, opening up promising avenues for future research.

Chapter 15

Outlook

This chapter discusses future work based on the presented approaches that can potentially further improve the performance and applicability of our work. We motivate additional studies to characterize adaptation planning strategies in general, highlight opportunities to improve the self-aware optimization framework as a whole, and discuss the transfer of research results from the systems-of-systems domain to the framework. Further, we propose research directions for our example use cases platooning coordination, rVRP, and mezzanine warehouses.

General Characterization of Adaptation Planning Strategies.

We examined various characteristics of adaptation planning strategies with a particular focus on ITS and logistics. Our case studies revealed the characteristics situation-awareness, fairness, and uncertainty. However, we do not claim the completeness of the identified characteristics in adaptation planning strategies, as we limited our research to the domains of ITS and logistics. Therefore, additional studies characterizing adaptation planning strategies in other use cases are useful to discuss the applicability of the framework in other domains and to derive possible further advancements of the framework.

Enhancing the Components of the Framework.

In this thesis, we proposed a self-aware optimization framework for adaptation planning strategies consisting of four components: (i) Coordination, (ii) Situation Detection, (iii) Strategy Selection, and (iv) Parameter Optimization. Each of these components has individual limitations that can be reduced in the future. First, the coordination component processes the observations from the application and triggers the other components. However, with increasing runtime of the framework, the amount of data collected from the application increases. This leads to large data sets that may become outdated [MS88, SLMS20]. Hence, a strategy on how to discard or aggregate the increasing amount of data could enhance the component. Further, the situation detection currently comprises a rule-based and a clustering approach, but it does not adapt the rule set with learned insights. Hence, a rule-learning mechanism could improve the rule base of the situation detection. Currently, the strategy selection learns which strategy to choose based solely on all observations of the current situation. However, a global mechanism could provide benefits to the component by adjusting the order of strategies based on the performance of strategies previously experienced in all situations. This could reduce the trial-and-error phase for new situations and shorten the convergence time. The parameter optimization component currently provides the hypervolume metric and individual thresholds. However, for other use cases, other techniques for multi-objective optimization could be useful, such as the concept of Pareto-optimality to provide the operator with a set of equally well performing configurations. Further, approaches to reduce the search space for parameter tuning, such as [PG18,HHLB11], could speed up the component or fitness landscape analysis could be applied to kick-start the optimization process [PA12]. In general, the component-based architecture of our framework allows for enhancement of each component according to the individual requirements and best practices of the targeted use case.

Transfer of Results in Optimizing Systems-of-Systems to the Framework.

At the current stage of the self-aware optimization framework, we did not integrate the approaches for tackling more complex systems-of-systems such as those for rVRP and processes in mezzanine warehouses. Currently, the framework can be applied to these use cases to identify the situation, select promising optimization approaches, and tune their parameters. However, it does not explicitly consider mutual interactions of the systems-of-systems structure, including vertical and horizontal systems. In the future, the framework could be extended to include mechanisms to deal with such complex systems and meet the specific requirements in such scenarios.

Advancing the Research on Platooning Coordination.

In this thesis, we used platooning coordination as a use case and applied three commonly used platooning coordination strategies. Further, we proposed six platooning sequence rotation mechanisms that can be used to balance positive and negative effects of platooning. Many approaches for platooning coordination have been proposed in the literature [LBS$^+$21]. Although many approaches proved their effectiveness in simulations, there is currently no quantitative comparison. In a recent publication, we presented a testbed [KLP$^+$19] based on the platooning simulator PLEXE [SJB$^+$14, Seg16]. However, this assessment is out of the scope of this thesis, but it can be a starting point for an in-depth comparison of approaches. Additionally, the definition of individual platooning coordination approaches that take into account individual goals and constraints as well as vehicular characteristics indicates another future direction.

Enhancement of the Optimization Workflow for rVRP.

In our contribution on optimization of nested systems-of-systems, we used the rVRP as a use case and applied GA and ACO as optimization algorithms. In the future, an investigation of other common optimization algorithms such as particle swarm or branch-and-bound algorithms within our two-stage approach is meaningful. Since hybrid approaches seem to be useful for such complex problems in the literature, an investigation of a mixture of GA, ACO, and LS at the different stages could be beneficial. Also, a multi-objective representation of the problem might be possible by transferring the constraints such as time window adjustment, reduction of required time, and travel distances as opposing objectives. Then we could apply common multi-objective optimization techniques such as NSGA-II and evaluate their performance.

Enhancement of Storage Assignment and Order Picking in Mezzanine Warehouses.

Finally, we studied coexisting systems-of-systems using the example of processes in mezzanine warehouses. Here also other optimization algorithms can be examined for this problem. Further, by integrating additional features, the applicability of storage assignment and order picking approaches can be enhanced. In this work, we assumed that the mezzanine warehouse does not change while we execute the optimization algorithms. However, since these systems are highly dynamic, accounting for state changes within the mezzanine warehouses while running the algorithms could lead to a more robust approach for real-world applications. Further, parallelizing the algorithms of both coexisting systems could reduce the required computation time. Finally, integrating forecasts of future assignment and order picking tasks could lead to a novel warehouse handling, as goods that are expected to be ordered in the near future could be relocated in advance towards the delivery point.

Appendices

Appendix A
Appendix

A.1 Full Specification of the Domain Data Model of the Self-Aware Optimization Framework

The following tables fully specify the DDM used in Chapter 7 to define use-case specific characteristics. Table A.1 defines the use case part of the DDM, Table A.2 specifies the context part, Table A.3 describes the parameter optimization part, and Table A.4 determines the performance metrics part of the DDM.

Table A.1: Definition of the Use Case part of the DDM.

Use Case		
Name	Type	Required
name	String	Yes
available_algos	List of String	Yes
fallback_rules	String	Yes

Table A.2: Definition of the Context part of the DDM.

Context		
Name	Type	Required
-	(`context:data`, `context:situation_detection_settings`)	Yes
data	List of `context:observation`	Yes
observation	(`context:name`, `context:type`)	Yes
name	String	Yes
type	int, double	Yes
situation_detection-_settings	(`context:algorithm`, `context:settings`)	Yes
algorithm	ENUM: kMeans, DBSCAN, OPTICS, RuleBased	Yes
settings	(`context:find_k`, `context:min_k`, `context:max_k`, `context:k`)	`if context:algorithm == kMeans`
	(`context:min_samples`, `context:eps`)	`if context:algorithm == DBSCAN`
	(`context:min_samples`, `context:min_cluster_size`)	`if context:algorithm == OPTICS`
	(`context:rule_set`)	`if context:algorithm == RuleBased`
find_k	Boolean	`if context:algorithm == kMeans`
min_k	int	`if context:algorithm == kMeans`
max_k	int	`if context:algorithm == kMeans`
k	int	`if context:algorithm == kMeans`
min_samples	int	`if context:algorithm == DBSCAN OR OPTICS`
eps	int	`if context:algorithm == DBSCAN`
min_cluster_size	int	`if context:algorithm == OPTICS`
rule_set	String	`if context:algorithm == RuleBased`

Table A.3: Definition of the Parameter Options part of the DDM.

Parameter Options (param_opt)		
Name	Type	Required
-	(`param_opt:data`, `param_opt:strategy_selection_settings`)	Yes
data	List of `param_opt:params`	Yes
params	(`param_opt:name`, `param_opt:algorithms`, `param_opt:data_type`, `param_opt:min`, `param_opt:max`, `param_opt:step`)	Yes
name	String	Yes
algorithms	List of Strings	No
data_type	int, double	Yes
min	int	Yes
max_k	int	Yes
step	int	No
strategy_selection-_setting	(`param_opt:min_optimization_attempts`, `param_opt:window_size`, `param_opt:threshold_exceeds`, `param_opt:method`, `param_opt:hypervolume_threshold`)	Yes
min_optimization-_attempts	int	Yes
window_size	int	Yes
threshold_exceeds	int	Yes
method	hypervolume, threshold	Yes
hypervolume_threshold	double	`if param_opt:method == hypervolume`

Table A.4: Definition of the Performance Metrics part of the DDM.

Performance Metrics (perf_metr)		
Name	Type	Required
-	List of `perf_metr:metrics`	Yes
metrics	(`perf_metr:data_type`, `perf_metr:higher_is_better`, `perf_metr:reference_value`, `perf_metr:threshold_value`)	Yes
data_type	int, double	Yes
higher_is_better	Boolean	Yes
reference_value	double	Yes
threshold_value	double	`if param_opt:method == threshold`

List of Figures

List of Tables

Acronyms

AC	Autonomic Computing.
ACC	Adaptive Cruise Control.
ACO	Ant Colony Optimization.
AMTICS	Advanced Mobile Traffic Information and Communication System.
API	Application Programming Interface.
APTS	Advanced Public Transportation Systems.
ARTS	Advanced Rural Transportation Systems.
ATIS	Advanced Traveler Information Systems.
ATMS	Advanced Traffic Management Systems.
AVCS	Advanced Vehicle Control Systems.
BD	Best Distance.
BDL	Best Distance and Lane.
BT	Belgian Tourniquet.
BTJS	Belgian Tourniquet Jump-start.
BV	Best Velocity.
C	Coverage.
C-VRP	Capacitated Vehicle Routing Problem.
CACC	Cooperative Adaptive Cruise Control.
CPS	Cyber-Physical System.
CVO	Commercial Vehicle Operations.

DAS	Dynamically Adaptive System.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise.
DDM	Domain Data Model.
DtB	Drafting a Single Vehicle to the Back.
DtF	Drafting a Single Vehicle to the Front.
ED	Euclidean Distance.
EGRS	Electronic Route Guidance System.
EUREKA	European Road Transport Telemetric Implementation Coordination Organization.
FADSE	Framework for Automatic Design Space Exploration.
GA	Genetic Algorithm.
GD	Generational Distance.
GS	Generated Spread.
GUI	Graphical User Interface.
HDV	Heavy-duty Vehicle.
HV	Hypervolume.
ICT	Information and Communications Technology.
IGD	Inverted Generational Distance.
ILP	Integer Linear Program.
IoT	Internet of Things.
ITS	Intelligent Transportation Systems.
KVM	Kernel-based Virtual Machines.
LRA-M Loop	Learn-Reason-Act-Model Loop.

LS Local Search.

LSP Logistic Service Providers.

MAPE-K Control Loop Monitor Analyze Plan Execute Knowledge Control Loop.

MDP Markov Decision Processes.

MMAS Max-Min Ant System.

MPC Model Predictive Control.

NSGA-II Non-dominated Sorting Genetic Algorithm II.

OBD On-Board Diagnostics.

OC Organic Computing.

OPTICS Ordering Points To Identify the Clustering Structure.

P&D Pickup and Delivery.

PCS Platooning Coordination System.

PD Proportional Derivative.

PFS Pareto Front Size.

PLEXE Platooning Extension for Veins.

PSP Platooning Service Providers.

RACS Road Automobile Communication System.

RBT Reversed Belgian Tourniquet.

RBTJS Reversed Belgian Tourniquet Jump-start.

REST Representational State Transfer.

rVRP Rich Vehicle Routing Problem.

SAS Self-adaptive System.

SeAC	Self-aware Computing.
SMBO	Sequential Model-based Bayesian Optimization.
SPEA2	Strength Pareto Evolutionary Algorithm.
SUMO	Simulation of Urban MObility.
TSP	Traveling Salesman Problem.
TW	Time Windows.
Veins	Vehicles in Network Simulation.
VM	Virtual Machine.
VRP	Vehicle Routing Problem.
VRP-TW-PD	Vehicle Routing Problem with Time Windows and Pickup and Delivery.
VRP-PD	Vehicle Routing Problem with Pickup and Delivery.
VRP-TW	Vehicle Routing Problem with Time Windows.
YAML	YAML Ain't Markup Language.

Bibliography

[Aa06] Fermín Alfredo Tang Montané and Roberto Diéguez Galv ao. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33(3):595–619, 2006. [see pages 77 and 251]

[AG] PTVPlanung Transport Verkehr AG. Regionaler Nahverkehrsplan Mittleres Mecklenburg/Rostock. `https://www.planungsverband-rostock.de/wp-content/uploads/2018/07/NVP%5F%5Fbersicht.pdf`. Last Accessed: 2021-10-07. [see page 179]

[AGM15] Alan Ali, Gaetan Garcia, and Philippe Martinet. The Flatbed Platoon Towing Model for Safe and Dense Platooning on Highways. *Intelligent Transportation Systems Magazine*, 7(1):58–68, 2015. [see page 33]

[AHM13] Samira Almoustafa, Said Hanafi, and Nenad Mladenović. New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research*, 226(3):386–394, 2013. [see pages 77 and 251]

[Ala11] Assad Alam. *Fuel-efficient distributed control for heavy duty vehicle platooning*. PhD thesis, KTH Royal Institute of Technology, Stockholm, 2011. [see page 3]

[ALS11] Sheng-hai An, Byung-Hyug Lee, and Dong-Ryeol Shin. A Survey of Intelligent Transportation Systems. In *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, pages 332–337. IEEE, 2011. [see page 27]

[AME$^+$09] Anant Agarwal, Jason Miller, Jonathan Eastep, David Wentziaff, and Harshad Kasture. Self-aware computing. Technical report, Massachusetts Institute of Technology, 2009. [see page 244]

[ASG07] I. Alaya, C. Solnon, and K. Ghedira. Ant Colony Optimization for Multi-Objective Optimization Problems. In *In Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 450–457, 2007. [see pages 171, 172, and 173]

[ATD$^+$21] Mahdi Alinaghian, Erfan Babaee Tirkolaee, Zahra Kaviani Dezaki, Seyed Reza Hejazi, and Weiping Ding. An augmented Tabu search algorithm for the green inventory-routing problem with time windows. *Swarm and Evolutionary Computation*, 60:100802, 2021. [see page 253]

[ATL14] Salem Alelyani, Jiliang Tang, and Huan Liu. Feature Selection for Clustering: A Review. *Data Clustering: Algorithms and Applications*, 29(110-121), 2014. [see page 106]

[BA03] Barrie M. Baker and M.A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003. [see pages 77 and 251]

[bas] bast (Bundesanstalt für Straßenwesen) - Automatische Zählstellen 2018. `https://www.bast.de/BASt`$_2$`017/DE/Verkehrstechnik/Fachthemen/v2-verkehrszaehlung/Daten/2018`$_1$`/Jawe2018.html`. Last Accessed: 2021-11-12. [see page 178]

[BB03] Jean Berger and Mohamed Barkaoui. A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem. In *Genetic and Evolutionary Computation — GECCO 2003*, pages 646–656, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. [see pages 77 and 251]

[BBD$^+$21] Tim Bolender, Gereon Bürvenich, Manuela Dalibor, Bernhard Rumpe, and Andreas Wortmann. Self-Adaptive Manufacturing with Digital Twins. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 156–166, 2021. [see pages 2 and 3]

[BCDF10] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010. [see page 61]

[BDMK21] Partha Sarathi Barma, Joydeep Dutta, Anupam Mukherjee, and Samarjit Kar. A multi-objective ring star vehicle routing problem for perishable items. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–26, 2021. [see page 252]

[BDSH13] Lakshmi Dhevi Baskar, Bart De Schutter, and Hans Hellendoorn. Optimal routing for automated highway systems. *Transportation Research Part C: Emerging Technologies*, 30:1–22, 2013. [see pages 41, 42, and 45]

[BGT$^+$19] Amel Bennaceur, Carlo Ghezzi, Kenji Tei, Timo Kehrer, Danny Weyns, Radu Calinescu, Schahram Dustdar, Zhenjiang Hu, Shinichi Honiden, Fuyuki Ishikawa, Zhi Jin, Jeffrey Kramer, Marin Litoiu, Michele Loreti, Gabriel Moreno, Hausi Müller, Laura Nenzi, Bashar Nuseibeh, Liliana Pasquale, Wolfgang Reisig, Heinz Schmidt, Christos Tsigkanos, and Haiyan Zhao. Modelling and Analysing Resilient Cyber-Physical Systems. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 70–76, 2019. [see page 2]

[BHMK11] Thomas Bousonville, Alexandra Hartmann, Teresa Melo, and Herbert Kopfer. Vehicle Routing and Refueling: The Impact of Price Variations on Tour Length. *Herausforderungen, Chancen und Lösungen Band II*, page 83, 2011. [see page 71]

[BHT12] Christian Becker, Jörg Hähner, and Sven Tomforde. Flexibility in Organic Systems - Remarks on Mechanisms for Adapting System Goals at Runtime. In *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, pages 287–292. SciTePress, 2012. [see page 248]

[BIH19] John J Bartholdi III and Steven Todd Hackman. *Warehouse and Distribution Science*. Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2019. [see pages 54, 55, 221, and 254]

[BJ15] Bart Besselink and Karl Henrik Johansson. Control of platoons of heavy-duty vehicles using a delay-based spacing policy. In *12th IFAC Workshop on Time Delay Systems (TDS 2015)*, pages 364–369, Ann Arbor, MI, 6 2015. Elsevier. [see page 33]

[BJ17] Bart Besselink and Karl Henrik Johansson. String Stability and a Delay-Based Spacing Policy for Vehicle Platoons Subject to Disturbances. *IEEE Transactions on Automatic Control*, 62(9):4376–4391, 9 2017. [see page 33]

[BKK$^+$16] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016. [see pages 5 and 244]

[BLV$^+$19] André Bauer, Veronika Lesch, Laurens Versluis, Alexey Ilyushkin, Nikolas Herbst, and Samuel Kounev. Chamulteon: Coordinated Auto-Scaling of Micro-Services. In *Proceedings of the 39th IEEE International Conference on Distributed Computing Systems*, July 2019. [see page xxv]

[BM04] John E. Bell and Patrick R. McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48, 2004. [see pages 77 and 251]

[Bre] Jörg Breithut. A8 zwischen Stuttgart und Leonberg: Polizei stellt Autobahn-Blitzer wieder auf. `https://www.stuttgarter-nachrichten.de/inhalt.a8-zwischen-stuttgart-und-leonberg-polizei-stellt-autobahn-blitzer-wieder-auf.631561fb-8f7f-4881-a4cc-74eac1f4a158.html`. Last Accessed: 2021-10-06. [see page 178]

[Bro11] Jason Brownlee. *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, 2011. [see pages 23, 24, and 25]

[BS03] Benjamin Baran and Matilde Schaerer. A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows. In *IASTED International Multi-Conference on Applied Informatics*, volume 21, pages 97–102, 01 2003. [see pages 77 and 251]

[BSC$^+$12] Carl Bergenhem, Steven Shladover, Erik Coelingh, Christoffer Englund, and Sadayuki Tsugawa. Overview of Platooning Systems. In *Proceedings of the 19th ITS World Congress*, 2012. [see pages 59, 67, 114, and 118]

[BT83] Egon Balas and Paolo Toth. Branch and Bound Methods for the Traveling Salesman Problem. 1983. [see page 50]

[BTASG19] Erfan Babaee Tirkolaee, Parvin Abbasian, Mehdi Soltani, and Seyed Ali Ghaffarian. Developing an applied algorithm for multi-trip vehicle routing problem with time windows in urban waste collection: A case study. *Waste Management & Research*, 37(1_suppl):4–13, 2019. [see page 252]

[BTGPMK19] Erfan Babaee Tirkolaee, Alireza Goli, Maryam Pahlevan, and Ramina Malekalipour Kordestanizadeh. A robust bi-objective multi-trip periodic capacitated arc routing problem for urban waste collection using a multi-objective invasive weed optimization. *Waste Management & Research*, 37(11):1089–1101, 2019. [see page 253]

[BTV$^+$16] Bart Besselink, Valerio Turri, Sebastian H. Van De Hoef, Kuo Yun Liang, Assad Alam, Jonas Mårtensson, and Karl H. Johansson. Cyber-Physical Control of Road Freight Transport. *Proceedings of the IEEE*, 104(5):1128–1141, 2016. [see pages 40, 41, and 45]

[BTW14] K. Bellman, S. Tomforde, and R. P. Würtz. Interwoven Systems: Self-Improving Systems Integration. In *IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 123–127, 2014. [see pages 5 and 249]

[bus] bussgeldkatalog.org. Wissenswertes zur Verkehrsstatistik für die Jahre 2018/2019. `https://www.bussgeldkatalog.org/verkehrsstatistik/#auch`$_i$`n`$_p$`uncto`$_s$`tau\`$_z$`eigt`$_d$`ie`$_s$`tatistik`$_e$`ine`$_s$`teigerung`$_i$`m`$_j$`ahr`$_2$`018`. Last Accessed: 2021-10-24. [see page 2]

[Çat09] Bülent Çatay. *Ant Colony Optimization and Its Application to the Vehicle Routing Problem with Pickups and Deliveries*, pages 219–244. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. [see pages 77, 145, and 251]

[CBK$^+$17] Javier Cámara, Kirstie L Bellman, Jeffrey O Kephart, Marco Autili, Nelly Bencomo, Ada Diaconescu, Holger Giese, Sebastian Götz, Paola Inverardi, Samuel Kounev, et al. Self-Aware Computing Systems: Related Concepts and Research Areas.

In *Self-Aware Computing Systems*, pages 17–49. Springer, 2017. [see page 86]

[CC02] Zbigniew J. Czech and P. Czarnas. Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 376–383, 2002. [see pages 77 and 251]

[CCZL19] Jinlong Chai, Jiangeng Chang, Yakun Zhao, and Honggang Liu. An Auto-ML Framework Based on GBDT for Lifelong Learning. *arXiv preprint arXiv:1908.11033*, 2019. [see page 245]

[CdLG$^+$09] Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee. *Software Engineering for Self-Adaptive Systems: A Research Roadmap*. Springer Berlin Heidelberg, 2009. [see pages 1, 5, and 17]

[CEE16] Jhon Jairo Santa Chávez, John Willmer Escobar, and Mauricio Granada Echeverri. A multi-objective Pareto ant colony algorithm for the Multi-Depot Vehicle Routing problem with Backhauls. *International Journal of Industrial Engineering Computations*, pages 35–48, 2016. [see pages 77 and 251]

[CGMS17] Javier Cámara, David Garlan, Gabriel A. Moreno, and Bradley Schmerl. Evaluating Trade-Offs of Human Involvement in Self-Adaptive Systems. In *Managing Trade-Offs in Adaptable Software Architectures*, pages 155–180. Morgan Kaufmann, 2017. [see page 248]

[CHD07] Ping Chen, Houkuan Huang, and Xingye Dong. An Ant Colony System Based Heuristic Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pickup. In *In Proceedings of the 2nd IEEE Conference on Industrial Electronics and Applications*, pages 136–141. IEEE, may 2007. [see pages 77 and 251]

[CHS$^+$18] Shelvin Chand, Quang Huynh, Hemant Singh, Tapabrata Ray, and Markus Wagner. On the Use of Genetic Programming to Evolve Priority Rules for Resource Constrained Project Scheduling Problems. *Information Sciences*, 432:146–163, 2018. [see page 106]

[CLSV07] Jean-François Cordeau, Gilbert Laporte, Martin W.P. Savelsbergh, and Daniele Vigo. Chapter 6 Vehicle Routing. In *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367–428. Elsevier, 2007. [see pages 77 and 251]

[CM12] Jean-François Cordeau and Mirko Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, 39(9):2033–2050, 2012. [see pages 77 and 251]

[CM13] Padmabati Chand and JR Mohanty. A Multi-Objective Vehicle Routing Problem using Dominant Rank Method. *International Journal of Computer Application*, pages 29–34, 2013. [see pages 77 and 251]

[CMG15] Javier Cámara, Gabriel A. Moreno, and David Garlan. Reasoning about Human Participation in Self-Adaptive Systems. In *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 146–156, 2015. [see page 248]

[CMMF17] Z. Al Chami, H. Manier, M.-A. Manier, and C. Fitouri. A hybrid genetic algorithm to solve a multi-objective Pickup and Delivery Problem. *IFAC-PapersOnLine*, 50(1):14656–14661, 2017. [see pages 77 and 251]

[CMPPW20] Radu Calinescu, Raffaela Mirandola, Diego Perez-Palacin, and Danny Weyns. Understanding Uncertainty in Self-adaptive Systems. In *In Proceedings of IEEE International Conference on Autonomic Computing and Self-Organizing Systems*, pages 242–251. IEEE, 2020. [see pages 5 and 243]

[Cox05] Michael T Cox. Metacognition in computation: A selected research review. *Artificial Intelligence*, 169:104–141, 2005. [see page 244]

[ÇS19] Melih Çelik and Haldun Süral. Order picking in parallel-aisle warehouses with multiple blocks: complexity and a graph theory-based heuristic. *International Journal of Production Research*, 57(3):888–906, 2019. [see page 74]

[CT10] Chia-Ho Chen and Ching-Jung Ting. Applying Two-Stage Ant Colony Optimization to Solve the Large Scale Vehicle Routing Problem. *Journal of the Eastern Asia Society for Transportation Studies*, 8:761–776, 2010. [see pages 77 and 131]

[CVV13] Radu Chis, Maria Vintan, and Lucian Vintan. Multi-objective DSE algorithms' evaluations on processor optimization. In *In Proceedings of the 9th International Conference on Intelligent Computer Communication and Processing*, pages 27–33. IEEE, 2013. [see pages 5 and 245]

[CW64] Geoff Clarke and John W Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, 1964. [see pages 51 and 198]

[CWQX13] Fangyu Chen, Hongwei Wang, Chao Qi, and Yong Xie. An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers & Industrial Engineering*, 66(1):77–85, 2013. [see pages 170 and 254]

[DBM$^+$20] Joydeep Dutta, Partha Sarathi Barma, Anupam Mukherjee, Samarjit Kar, and Tanmay De. A multi-objective open set orienteering problem. *Neural Computing and Applications*, 32(17):13953–13969, 2020. [see page 252]

[DCH08] Thanh-Son Dao, Christopher Michael Clark, and Jan Paul Huissoon. Distributed Platoon Assignment and Lane Selection for Traffic Flow Optimization. In *Proceedings of the 2008 IEEE Intelligent Vehicles Symposium*, pages 739–744, 2008. [see page 39]

[DDE$^+$02] Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius M. Solomon, and François Soumis. 9. VRP with Pickup and Delivery. In *The Vehicle Routing Problem*, pages 225–242. Society for Industrial and Applied Mathematics, jan 2002. [see page 251]

[DFJ54] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954. [see page 50]

[DG97] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73–81, 1997. [see page 50]

[DHR00] Karl Doerner, Richard F. Hartl, and Marc Reimann. Ant colony optimization applied to the pickup and delivery problem. Technical Report 76, SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, Vienna, 2000. [see pages 77 and 251]

[DK07] M B M. De Koster. Warehouse Assessment in a Single Tour. In *Facility Logistics*, pages 53–74. Auerbach Publications, 2007. [see pages 4, 52, and 74]

[DKLDR07] René De Koster, Tho Le-Duc, and Kees Jan Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501, 2007. [see pages 4, 52, 55, 74, 222, and 254]

[DMC96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):29–41, 1996. [see pages 25 and 141]

[dpa] dpa/lsw. Verkehr - Stuttgart - Meistbefahrener Autobahnabschnitt: Unfallzahlen verdoppelt - Wirtschaft - SZ.de. `https://www.sueddeutsche.de/wirtschaft/verkehr-stuttgart-meistbefahrener-autobahnabschnitt-unfallzahlen-verdoppelt-dpa.urn-newsml-dpa-com-20090101-170806-99-537657`. Last Accessed: 2021-10-06. [see page 177]

[DPAM02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. [see pages 25, 61, 154, and 162]

[DPH17] Victor S. Dolk, Jeroen Ploeg, and W. P. Maurice H. Heemels. Event-Triggered Control for String-Stable Vehicle Platooning. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3486–3500, 9 2017. [see page 33]

[DPW09] Ding-Zhu Du, Panos M. Pardalos, and Weili Wu. *History of Optimization*, pages 1538–1542. Springer US, Boston, MA, 2009. [see page 19]

[DR59] George B Dantzig and John H Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 1959. [see page 50]

[DRS98] Richard L Daniels, Jeffrey L Rummel, and Robert Schantz. A model for warehouse order picking. *European Journal of Operational Research*, 105(1):1–17, 1998. [see page 254]

[DRTR$^+$19] Floriano De Rango, Mauro Tropea, Pierfrancesco Raimondo, Amilcare Francesco Santamaria, and Peppino Fazio. Bio Inspired Strategy for Improving Platoon Management in the Future Autonomous Electrical VANET Environment. In *Proceedings of the 28th IEEE International Conference on Computer Communication and Networks*, 2019. [see page 45]

[DRTRS20] Floriano De Rango, Mauro Tropea, Pierfrancesco Raimondo, and Amilcare Francesco Santamaria. Grey Wolf Optimization in VANET to manage Platooning of Future Autonomous Electrical Vehicles. In *Proceedings of the IEEE 17th Annual Consumer Communications & Networking Conference*, 2020. [see page 45]

[DX06] Lin Wei Dong and Cai Tian Xiang. Ant Colony Optimization for VRP and Mail Delivery Problems. In *In Proceedings of the 4th IEEE International Conference on Industrial Informatics*, pages 1143–1148, 2006. [see pages 77 and 251]

[DYW$^+$16] Kakan C. Dey, Li Yan, Xujie Wang, Yue Wang, Haiying Shen, Mashrur Chowdhury, Lei Yu, Chenxi Qiu, and Vivekgautham Soundararaj. A Review of Communication, Driver Characteristics, and Controls Aspects of Cooperative Adaptive Cruise Control (CACC). *IEEE Transactions on Intelligent Transportation Systems*, 17(2):491–509, 2 2016. [see page 33]

[Eil15] Sönke Eilers. COMPANION Deliverable D3.2 - Information Model for Platoon Services. Technical Report 610990, 2015. [see page 45]

[EKM11] Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek. Taming Uncertainty in Self-adaptive Software. In *Proceedings of the 19th*

ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, pages 234–244, 2011. [see page 248]

[EM13] Naeem Esfahani and Sam Malek. *Uncertainty in Self-Adaptive Software Systems*, pages 14—-238. Springer, 2013. [see pages 71 and 249]

[End17] Mica R Endsley. Toward a Theory of Situation Awareness in Dynamic Systems. In *Human Factors: The Journal of Human Factors and Ergonomics Society*, volume 37, pages 32–64. Sage Journals, 2017. [see pages 5 and 243]

[ENOBTMG16] David Espinoza-Nevárez, José Carlos Ortiz-Bayliss, Hugo Terashima-Marín, and Gustavo Gatica. Selection and Generation Hyper-Heuristics for Solving the Vehicle Routing Problem with Time Windows. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, page 139–140, 2016. [see pages 77 and 251]

[EÖ12] Seval Ene and Nursel Öztürk. Storage location assignment and order picking optimization in the automotive industry. *The International Journal of Advanced Manufacturing Technology*, 60:787–797, 2012. [see page 254]

[EU221] Directive 2010/40/EU of the European Parliament and of the Council of 7 July 2010 on the framework for the deployment of Intelligent Transport Systems in the field of road transport and for interfaces with other modes of transport Text with EEA relevance. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32010L0040`, May 2021. [Online; accessed 28. May 2021]. [see page 27]

[FGKV19] Erik M Fredericks, Ilias Gerostathopoulos, Christian Krupitzer, and Thomas Vogel. Planning as Optimization: Dynamically Discovering Optimal Configurations for Runtime Situations. In *In Proceedings of the 13th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 1–10. IEEE, 2019. [see pages 5, 106, and 243]

[FJM$^+$01] Lino Figueiredo, Isabel Jesus, JA Tenreiro Machado, Jose Rui Ferreira, and JL Martins De Carvalho. Towards the development of intelligent transportation systems. In *In Proceedings of*

IEEE Intelligent Transportation Systems, pages 1206–1211. IEEE, 2001. [see pages 2 and 27]

[FKH11] Camilo Fitzgerald, Benjamin Klöpper, and Shinichi Honiden. Utility-Based Self-Adaption with Environment Specific Quality Models. In Abdelhamid Bouchachia, editor, *Adaptive and Intelligent Systems*, pages 107–118. Springer, 2011. [see page 249]

[FLL$^+$05] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F. Werneck. Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3):491–511, oct 2005. [see pages 77 and 251]

[Flo56] Merrill M Flood. The Traveling-Salesman Problem. *Operations research*, 4(1):61–75, 1956. [see page 50]

[FMP07] Daniela Favaretto, Elena Moretti, and Paola Pellegrini. Ant colony system for a VRP with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10:263–284, 04 2007. [see pages 77 and 251]

[FN15] Pedro Fernandes and Urbano Nunes. Multiplatooning Leaders Positioning and Cooperative Behavior Algorithms of Communicant Automated Vehicles for High Traffic Capacity. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1172–1187, 2015. [see page 42]

[Fra02] Edward Frazelle. *Supply Chain Strategy: The Logistics of Supply Chain Management*. McGrraw-Hill, 2002. [see page 55]

[FSH15] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. [see pages 5 and 245]

[FTV94] Matteo Fischetti, Paolo Toth, and Daniele Vigo. A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs. *Operations Research*, 42(5):846–859, oct 1994. [see pages 77 and 251]

[FVDW21] Nicola Franco, Hoai My Van, Marc Dreiser, and Gereon Weiss. Towards a Self-Adaptive Architecture for Federated Learning of Industrial Automation Systems. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 210–216, 2021. [see pages 2 and 3]

[FZL+19] Shuo Feng, Yi Zhang, Shengbo Eben Li, Zhong Cao, Henry X. Liu, and Li Li. String stability for vehicular platoon control: Definitions and analysis methods. *Annual Reviews in Control*, 47:81–97, 2019. [see page 31]

[GBBGP21] Mathieu Guillame-Bert, Sebastian Bruch, Josh Gordon, and Jan Pfeifer. Introducing TensorFlow Decision Forests. `https://blog.tensorflow.org/2021/05/introducing-tensorflow-decision-forests.html`, Nov 2021. [Online; Accessed 2. Nov. 2021]. [see page 110]

[GBH+17] Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, Adam Hujecek, Frantisek Plasil, and Dominik Skoda. Strengthening Adaptation in Cyber-Physical Systems via Meta-Adaptation Strategies. *ACM Transactions on Cyber-Physical Systems*, 1(3):1–25, 2017. [see page 244]

[GC09] Pablo Garrido and Carlos Castro. Stable Solving of CVRPs Using Hyperheuristics. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, page 255–262. ACM, 2009. [see pages 77 and 251]

[GDC+17] Julian Pedro Garbiso, Ada Diaconescu, Marceau Coupechoux, Jeremy Pitt, and Bertrand Leroy. Distributive Justice for Fair Auto-Adaptive Clusters of Connected Vehicles. In *2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pages 79–84, 2017. [see page 250]

[GG78] Bezalel Gavish and Stephen C Graves. The Travelling Salesman Problem and Related Problems. *Operations Research Center Working Papers*, 1978. [see page 51]

[GG10] Keivan Ghoseiri and Seyed Farid Ghannadpour. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing*, 10(4):1096–1107, sep 2010. [see pages 77 and 251]

[GGM10] Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis. Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539–549, 2010. [see pages 6, 75, and 147]

[Glo86] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986. [see page 198]

[GLP02] Michel Gendreau, Gilbert Laporte, and Jean-Yves Potvin. 6. Metaheuristics for the Capacitated VRP. In *The Vehicle Routing Problem*, pages 129–154. Society for Industrial and Applied Mathematics, jan 2002. [see pages 77 and 251]

[GMS$^+$09] Adam Ghandar, Zbigniew Michalewicz, Martin Schmidt, Thuy-Duong To, and Ralf Zurbrugg. Computational Intelligence for Evolving Trading Rules. *IEEE Transactions on Evolutionary Computation*, 13(1):71–86, 2009. [see page 106]

[Gov] Robert Gove. Using the elbow method to determine the optimal number of clusters for k-means clustering - bl.ocks.org. `https://bl.ocks.org/rpgove/0060ff3b656618e9136b`. Accessed: 2021-03-12. [see page 106]

[GP19] Ilias Gerostathopoulos and Evangelos Pournaras. TRAPPed in Traffic? A Self-Adaptive Framework for Decentralized Traffic Optimization. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 32–38, 2019. [see page 2]

[GPB18] Ilias Gerostathopoulos, Christian Prehofer, and Tomas Bures. Adapting a System with Noisy Outputs with Statistical Guarantees. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, page 58–68. ACM, 2018. [see page 249]

[GPFA16] Miriam Gil, Vicente Pelechano, Joan Fons, and Manoli Albert. Designing the Human in the Loop of Self-Adaptive Systems. In Carmelo R. García, Pino Caballero-Gil, Mike Burmester, and Alexis Quesada-Arencibia, editors, *Ubiquitous Computing and Ambient Intelligence*, pages 437–449. Springer, 2016. [see page 248]

[GRW08] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The Vehicle Routing Problem: Latest Advances and New*

Challenges, volume 43. Springer Science & Business Media, 2008. [see pages 5 and 76]

[GSBLC19] Giulia Giordano, Michele Segata, Franco Blanchini, and Renato Lo Cigno. The joint network/control design of platooning algorithms can enforce guaranteed safety constraints. *Ad Hoc Networks*, 94, 11 2019. [see page 33]

[GSCG17] Thomas J. Glazier, Bradley Schmerl, Javier Cámara, and David Garlan. Utility Theory for Self-Adaptive Systems. Technical Report CMU-ISR-17-119, CMU, 2017. [see page 249]

[HC05] Randolph Hall and Chinan Chin. Vehicle Sorting for Platoon Formation: Impacts on Highway Entry and Throughput. *Transportation Research Part C: Emerging Technologies*, 13(5-6):405–420, 2005. [see pages 41 and 42]

[Hes63] James L Heskett. Cube-per-order index-a key to warehouse stock location. *Transportation and Distribution Management*, 3(1):27–31, 1963. [see page 55]

[HGR$^+$21] Stefan Herrnleben, Johannes Grohmann, Pitor Rygielski, Veronika Lesch, Christian Krupitzer, and Samuel Kounev. A Simulation-based Optimization Framework for Online Adaptation of Networks. In Houbing Song and Dingde Jiang, editors, *Proceedings of the 12th EAI International Conference on Simulation Tools and Techniques*, SIMUtools 2020, page 513–532, Cham, August 2021. Springer International Publishing. [see page xxiv]

[HHLB11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. [see page 262]

[HI20] Timo Hintsch and Stefan Irnich. Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, 280(1):164–178, 2020. [see pages 77 and 251]

[HLL$^+$21] Stefan Herrnleben, Maximilian Leidinger, Veronika Lesch, Thomas Prantl, Johannes Grohmann, Christian Krupitzer, and

Samuel Kounev. ComBench: A Benchmarking Framework for Publish/Subscribe Communication Protocols under Network Limitations. In *Proceedings of the 13th EAI International Conference on Simulation Tools and Techniques*, SIMUtools 2021, 2021. [see page xxiv]

[HLS+21] Elia Henrichs, Veronika Lesch, Martin Straesser, Samuel Kounev, and Christian Krupitzer. A Literature Review on Optimization Techniques for Adaptation Planning in Adaptive Systems: State of the Art and Research Directions. *Information and Software Technology*, 2021. Under Review, Impact Factor (2020): 2.73. [see page xxiv]

[HM15] Shihong Huang and Pedro Miranda. Incorporating Human Intention into Self-Adaptive Systems. In *37th IEEE International Conference on Software Engineering*, pages 571–574, 2015. [see page 248]

[HMS+17] Abid Hussain, Yousaf Shad Muhammad, M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry, and Showkat Gani. Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. *Computational Intelligence and Neuroscience*, 2017:1–7, 2017. [see page 140]

[Hol92] John H. Holland. Genetic Algorithms. *Scientific American*, 267(1):66–73, 1992. [see pages 24 and 136]

[HRFA19] Claretha Hughes, Lionel Robert, Kristin Frady, and Adam Arroyos. Artificial intelligence, Employee Engagement, Fairness, and Job Outcomes. In *Managing Technology and Middle-and Low-skilled Employees*. Emerald Publishing Limited, 2019. [see page 75]

[HRMS98] William H Hendrix, Tina Robbins, Janis Miller, and Timothy P Summers. Effects of Procedural and Distributive Justice on Factors Predictive of Turnover. *Journal of Social Behavior and Personality*, 13(4):611, 1998. [see page 75]

[HS19a] Tobias Hardes and Christoph Sommer. Dynamic Platoon Formation at Urban Intersections. In *Proceedings of the 44th IEEE Conference on Local Computer Networks*, pages 101–104, 2019. [see pages 5, 39, and 45]

[HS19b] Tobias Hardes and Christoph Sommer. Towards Heterogeneous Communication Strategies for Urban Platooning at Intersections. In *2019 IEEE Vehicular Networking Conference*, pages 1–8. IEEE, 2019. [see page 244]

[HWW+17] Peng Hao, Ziran Wang, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew Barth. Intra-Platoon Vehicle Sequence Optimization for Eco-Cooperative Adaptive Cruise Control. In *IEEE 20th International Conference on Intelligent Transportation Systems*, 2017. [see page 247]

[ICMPG16] Joaquín Izquierdo, Enrique Campbell, Idel Montalvo, and Rafael Pérez-García. Injecting problem-dependent knowledge to improve evolutionary optimization search ability. *Journal of Computational and Applied Mathematics*, 291:281–292, 2016. [see page 150]

[ISR+19] Lukas Iffländer, Jonathan Stoll, Nishant Rawtani, Veronika Lesch, Klaus-Dieter Lange, and Samuel Kounev. Performance Oriented Dynamic Bypassing for Intrusion Detection Systems. In *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, ICPE '19, page 159–166, New York, NY, USA, 2019. ACM. [see page xxv]

[JCS+19] Pooyan Jamshidi, Javier Cámara, Bradley Schmerl, Christian Käestner, and David Garlan. Machine Learning Meets Quantitative Planning: Enabling Self-Adaptation in Autonomous Robots. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 39–50, 2019. [see pages 2 and 3]

[JFL+01] Nicholas R Jennings, Peyman Faratin, Alessio R Lomuscio, Simon Parsons, Michael J Wooldridge, and Carles Sierra. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, 10(2), 2001. [see page 250]

[JM03] David S Johnson and Lyle A McGeoch. 8. The traveling salesman problem: a case study. In *Local Search in Combinatorial Optimization*, pages 215–310. Princeton University Press, 2003. [see page 50]

[JNJM18] Alexander Johansson, Ehsan Nekouei, Karl Henrik Johansson, and Jonas Mårtensson. Multi-Fleet Platoon Matching: A

Game-Theoretic Approach. In *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems*, pages 2980–2985, 2018. [see page 45]

[Joo12] P Jootel. Safe road trains for the environment (SARTRE) final report. *tech. rep., Technical Report for European Commission under the Framework 7 Programme Project 233683)*, 2012. [see page 67]

[JZBdK15] Robbert Janssen, Han Zwijnenberg, Iris Blankers, and Janiek de Kruijff. Truck platooning: Driving the future of transportation. Technical report, TNO, 2015. [see page 247]

[KAFF14] Roozbeh Kianfar, Mohammad Ali, Paolo Falcone, and Jonas Fredriksson. Combined longitudinal and lateral control design for string stable vehicle platooning within a designated lane. In *17th International IEEE Conference on Intelligent Transportation Systems*, pages 1003–1008. IEEE, 10 2014. [see page 33]

[Kar72] Richard M Karp. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972. [see pages 50 and 51]

[KB05] Majid Ali Khan and Ladislau Bölöni. Convoy driving through ad-hoc coalition formation. In *Proceedings of the 11th IEEE Real Time and Embedded Technology and Applications Symposium*, pages 98–105, 2005. [see pages 40 and 42]

[KBW$^+$10] Monika Kofler, Andreas Beham, Stefan Wagner, Michael Affenzeller, and Clemens Reitinger. Reassigning storage locations in a warehouse to optimize the order picking process. In *Proceedings of the 22th European Modeling and Simulation Symposium*, pages 77–82, 2010. [see page 254]

[KC03] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50, 2003. [see pages 1, 2, 17, 46, 88, and 120]

[KCP20] Sehyeok Kang, Taeyeong Choi, and Theodore P Pavlic. How far should I watch? Quantifying the effect of various observational capabilities on long-range situational awareness in multi-robot teams. In *In Proceedings of the IEEE International*

Conference on Autonomic Computing and Self-Organizing Systems, pages 146–152. IEEE, 2020. [see pages 5, 107, and 244]

[KCW$^+$18] Cody Kinneer, Zack Coker, Jiacheng Wang, David Garlan, and Claire Le Goues. Managing Uncertainty in Self-Adaptive Systems with Plan Reuse and Stochastic Search. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, pages 40–50, 2018. [see pages 244 and 249]

[KFF11] Roozbeh Kianfar, Paolo Falcone, and Jonas Fredriksson. A Receding Horizon Approach to String Stable Cooperative Adaptive Cruise Control. In *Proceedings of the 14th International IEEE Intelligent Transportation Systems Conference*, pages 734–739, Washington, D.C., 10 2011. IEEE. [see page 32]

[KGV83] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. [see pages 24 and 61]

[KHNT19] Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation*, 27(1):3–45, 2019. [see pages 5 and 244]

[KKHT15] Lars Kotthoff, Pascal Kerschke, Holger Hoos, and Heike Trautmann. Improving the State of the Art in Inexact TSP Solving Using Per-Instance Algorithm Selection. In Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eléonore Marmion, editors, *Learning and Intelligent Optimization*, pages 202–217, Cham, 2015. Springer International Publishing. [see pages 5 and 244]

[KLB$^+$17] Samuel Kounev, Peter Lewis, Kirstie L Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, et al. The Notion of Self-aware Computing. In *Self-Aware Computing Systems*, pages 3–16. Springer, 2017. [see pages 1, 2, 17, 18, 82, and 88]

[KLP$^+$19] Christian Krupitzer, Veronika Lesch, Martin Pfannemüller, Christian Becker, and Michele Segata. A Modular Simulation Framework for Analyzing Platooning Coordination. In *Proceedings of the 1st ACM Workshop on Technologies, mOdels, and*

Protocols for Cooperative Connected Cars (TOP-Cars), Colocated with ACM MobiHoc 2019. ACM, July 2019. [see pages xxv, 46, 61, and 263]

[KLR$^+$20] Dennis Kaiser, Veronika Lesch, Julian Rothe, Michael Strohmeier, Florian Spiess, Christian Krupitzer, Sergio Montenegro, and Samuel Kounev. Towards Self-Aware Multirotor Formations. *Computers*, 9(7), Februar 2020. Special Issue on Self-Aware Computing. [see page xxiii]

[KM07] Jeff Kramer and Jeff Magee. Self-Managed Systems: an Architectural Challenge. In *In Proceedings of Future of Software Engineering*, pages 259–268. IEEE, 2007. [see pages 82 and 86]

[KML$^+$20] Christian Krupitzer, Sebastian Müller, Veronika Lesch, Marwin Züfle, Janick Edinger, Alexander Lemken, Dominik Schäfer, Samuel Kounev, and Christian Becker. A Survey on Human Machine Interaction in Industry 4.0. Technical report, Universität Würzburg and University of Mannheim and ioxp GmbH and Syntax Systems GmbH, feb 2020. [see page xxvi]

[Kor21] Martin Kords. Transportleistung im Straßengüterverkehr 2019 | Statista. `https://de.statista.com/statistik/daten/studie/2979/umfrage/entwicklung-der-transportleistung-des-strassengueterverkehrs`, 2021. [Online; acc. 3. Feb. 2021]. [see pages 3 and 76]

[Kov11] András Kovács. Optimizing the storage assignment in a warehouse served by milkrun logistics. *International Journal of Production Economics*, 133(1):312–318, 2011. [see page 254]

[KPP08] Konstantinos Kakousis, Nearchos Paspallis, and George A. Papadopoulos. Optimizing the Utility Function-Based Self-adaptive Behavior of Context-Aware Systems Using User Feedback. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008*, pages 657–674. Springer, 2008. [see page 249]

[KRV$^+$15] Christian Krupitzer, Felix Maximilian Roth, Sebastian Vansyckel, Gregor Schiele, and Christian Becker. A survey on engineering approaches for self-adaptive systems. *Pervasive and*

Mobile Computing, 17:184–206, 2015. [see pages 1, 5, 17, 46, 119, and 120]

[KSB$^+$18] Christian Krupitzer, Michele Segata, Martin Breitbach, Samy El-Tawab, Sven Tomforde, and Christian Becker. Towards Infrastructure-Aided Self-Organized Hybrid Platooning. In *In Proceedings of IEEE Global Conference on Internet of Things*, 2018. [see pages 29 and 46]

[KT05] Vijay R Kannan and Keah Choon Tan. Just in time, total quality management, and supply chain management: understanding their linkages and impact on business performance. *Omega*, 33(2):153–162, 2005. [see page 49]

[KT19] Pascal Kerschke and Heike Trautmann. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation*, 27(1):99–127, 03 2019. [see pages 5 and 244]

[KTSA14] V. Sivaram Kumar, M.R. Thansekhar, R. Saravanan, and S. Miruna Joe Amali. Solving Multi-objective Vehicle Routing Problem with Time Windows by FAGA. *Procedia Engineering*, 97:2176–2185, 2014. [see pages 77 and 251]

[KWZ$^+$20] Christian Krupitzer, Tim Wagenhals, Marwin Züfle, Veronika Lesch, Dominik Schäfer, Amin Mozaffarin, Janick Edinger, Christian Becker, and Samuel Kounev. A Survey on Predictive Maintenance for Industry 4.0. Technical report, Universität Würzburg and University of Mannheim and Syntax Systems GmbH and MOZYS Engineering GmbH, feb 2020. [see page xxvi]

[LAB$^+$11] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards Fully Autonomous Driving : Systems and Algorithms. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 163–168, 2011. [see page 31]

[LAG11] Kuo-Yun Liang, Assad Alam, and Ather Gattami. The impact of heterogeneity and order in heavy duty vehicle platooning networks (poster). In *2011 IEEE Vehicular Networking Conference*, pages 291–297, 2011. [see page 247]

[Lap92] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992. [see page 51]

[LBHK18] Veronika Lesch, André Bauer, Nikolas Herbst, and Samuel Kounev. FOX: Cost-Awareness for Autonomic Resource Management in Public Clouds. In *Proceedings of the 9th ACM/SPEC International Conference on Performance Engineering (ICPE 2018)*, New York, NY, USA, April 2018. ACM. [see page xxiv]

[LBL$^+$17] Peter Lewis, Kirstie L Bellman, Christopher Landauer, Lukas Esterle, Kyrre Glette, Ada Diaconescu, and Holger Giese. Towards a framework for the levels and aspects of self-aware computing systems. In *Self-Aware Computing Systems*, pages 51–85. Springer, 2017. [see page 244]

[LBS$^+$21] Veronika Lesch, Martin Breitbach, Michele Segata, Christian Becker, Samuel Kounev, and Christian Krupitzer. An Overview on Approaches for Coordination of Platoons. *IEEE Transactions on Intelligent Transportation Systems*, 2021. Impact Factor (2020): 6.492. [see pages xxiii, 29, 30, 34, and 263]

[LCF$^+$15] Peter R. Lewis, Arjun Chandra, Funmilade Faniyi, Kyrre Glette, Tao Chen, Rami Bahsoon, Jim Torresen, and Xin Yao. Architectural Aspects of Self-Aware and Self-Expressive Computing Systems: From Psychology to Engineering. *IEEE Computer*, 48(8):62–70, 2015. [see page 250]

[lCkYmW06] Ai ling Chen, Gen ke Yang, and Zhi ming Wu. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University-SCIENCE A*, 7(4):607–614, mar 2006. [see pages 77 and 251]

[LCL08] Meijuan Li, Xuebo Chen, and Chenqi Liu. Pareto and Niche Genetic Algorithm for Storage Location Assignment Optimization Problem. In *3rd International Conference on Innovative Computing Information and Control*, pages 465–465. IEEE, 2008. [see page 254]

[Les20] Veronika Lesch. Toward a Framework for Self-Learning Adaptation Planning through Optimization. In Sven Tomforde and Christian Krupitzer, editors, *Organic Computing: Doctoral Dissertation Colloquium 2020*, pages 17–31. Kassel University Press GmbH, jul 2020. [see pages xxvi, 1, 9, 17, 82, and 243]

[LHKK21a] Veronika Lesch, Marius Hadry, Samuel Kounev, and Christian Krupitzer. A Case Study on Optimization of Platooning Coordination. Technical report, Universität Würzburg, 2021. [see pages xxvi, 9, 82, and 177]

[LHKK21b] Veronika Lesch, Marius Hadry, Samuel Kounev, and Christian Krupitzer. Utility-based Vehicle Routing Integrating User Preferences. In *Proceedings of 3rd International Workshop on Pervasive Computing for Vehicular Systems, 2021*. IEEE, March 2021. [see pages xxv, 9, 10, 71, 72, 120, and 248]

[LHKK21c] Veronika Lesch, Marius Hadry, Christian Krupitzer, and Samuel Kounev. A Self-Aware Optimization Framework for Adaptation Planning Strategies. *ACM Transactions on Autonomous and Adaptive Systems*, 2021. In Preparation. [see pages xxiii, 9, 82, and 177]

[Lia14] Kuo-Yun Liang. *Coordination and Routing for Fuel-Efficient Heavy-Duty Vehicle Platoon Formation*. PhD thesis, KTH, Automatic Control, 2014. [see page 45]

[LJD$^+$17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017. [see page 245]

[LKK$^+$21a] Veronika Lesch, Maximilian König, Samuel Kounev, Anthony Stein, and Christian Krupitzer. Tackling the Rich Vehicle Routing Problem with Nature-Inspired Algorithms. *Applied Intelligence*, 2021. Accepted - In Press, Impact Factor (2020): 5.086. [see pages xxiii, 3, 9, 11, 76, 128, 195, and 251]

[LKK$^+$21b] Veronika Lesch, Maximilian König, Samuel Kounev, Anthony Stein, and Christian Krupitzer. A Case Study of Vehicle Route Optimization, 2021. [see pages xxvi, 11, 76, 128, and 195]

[LKPA15] Wei Liu, Seong-Woo Kim, Scott Pendleton, and Marcelo H Ang. Situation-aware decision making for autonomous driving on urban road using online POMDP. In *2015 IEEE Intelligent Vehicles Symposium*, pages 1126–1133. IEEE, 2015. [see pages 5 and 243]

[LKS^{+}21] Veronika Lesch, Christian Krupitzer, Kevin Stubenrauch, Nico Keil, Christian Becker, Samuel Kounev, and Michele Segata. A Comparison of Mechanisms for Compensating Negative Impacts of System Integration. *Future Generation Computer Systems*, 116:117–131, March 2021. [see pages xxiii, 3, 9, 10, 67, 114, 246, and 249]

[LKT19a] Veronika Lesch, Christian Krupitzer, and Sven Tomforde. Emerging Self-Integration through Coordination of Autonomous Adaptive Systems. In *Proceedings of the 4th IEEE International Workshops on Foundations and Applications of Self* Systems (FAS* W) 2019*. IEEE, June 2019. [see pages xxv and 250]

[LKT19b] Veronika Lesch, Christian Krupitzer, and Sven Tomforde. Multi-objective Optimisation in Hybrid Collaborating Adaptive Systems. In *Proceedings of the 7th edition in the Series on Autonomously Learning and Optimising Systems, co-located with 32nd GI/ITG ARCS 2019*. Gesellschaft fuer Informatik (GI), may 2019. [see pages xxv, 2, 3, 18, 28, 85, and 88]

[LL02] Hoong Chuin Lau and Zhe Liang. Pickup and delivery with time windows: Algorithms and test case generation. *International Journal on Artificial Intelligence Tools*, 11(03):455–472, 2002. [see pages 77 and 251]

[LLJ15] Jeffrey Larson, Kuo Yun Liang, and Karl H. Johansson. A Distributed Framework for Coordinated Heavy-Duty Vehicle Platooning. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):419–429, 2015. [see page 40]

[LMJ16] Kuo Yun Liang, Jonas Mårtensson, and Karl H. Johansson. Heavy-Duty Vehicle Platoon Formation for Fuel Efficiency. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1051–1061, 2016. [see page 45]

[LMK$^+$21a] Veronika Lesch, Patrick B. M. Müller, Moritz Krämer, Samuel Kounev, and Christian Krupitzer. Optimizing Storage Assignment, Order Picking, and their Interaction in Mezzanine Warehouses. *Applied Intelligence*, 2021. Under Review, Impact Factor (2020): 5.086. [see pages xxiv, 4, 9, 11, 74, 148, 219, and 254]

[LMK$^+$21b] Veronika Lesch, Patrick B.M. Müller, Moritz Krämer, Samuel Kounev, and Christian Krupitzer. A Case Study on Optimization of Warehouses. Technical report, 2021. Submitted to arxiv.org. [see pages xxvi, 11, 19, 52, 74, 148, and 219]

[LMS01] Helena R Lourenço, Olivier Martin, and Thomas Stützle. A beginner's introduction to Iterated Local Search. In *Proceedings of the 4th Metaheuristics International Conference*, volume 4, pages 1–11, 2001. [see page 24]

[LMS16] Jeffrey Larson, Todd Munson, and Vadim Sokolov. Coordinated Platoon Routing in a Metropolitan Network. In *Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*, pages 73–82, 2016. [see page 40]

[LN87] Gilbert Laporte and Yves Nobert. Exact Algorithms for the Vehicle Routing Problem. In Silvano Martello, Gilbert Laporte, Michel Minoux, and Celso Ribeiro, editors, *Surveys in Combinatorial Optimization*, North-Holland Mathematics Studies, pages 147–184. North-Holland, 1987. [see pages 51, 77, and 251]

[LNH$^+$21] Veronika Lesch, Tanja Noack, Johannes Hefter, Samuel Kounev, and Christian Krupitzer. Towards Situation-Aware Meta-Optimization of Adaptation Planning Strategies. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS 2021)*. IEEE, 2021. Best paper candidate. [see pages xxiv, 9, 61, 111, 180, 192, 193, and 243]

[LR98] Paul D Larson and Dale S Rogers. Supply Chain Management: Definition, Growth and Approaches. *Journal of Marketing Theory and Practice*, 6(4):1–5, 1998. [see page 49]

[LS08] Joel Lehman and Kenneth O Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *ALIFE*, pages 329–336, 2008. [see page 143]

[LS10] Joel Lehman and Kenneth O Stanley. Efficiently evolving programs through the search for novelty. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 837–844. ACM, 2010. [see page 61]

[LSG06] Adam N Letchford and Juan-José Salazar-González. Projection results for vehicle routing. *Mathematical Programming*, 105(2):251–274, 2006. [see page 51]

[LSL15] Erik Larsson, Gustav Sennton, and Jeffrey Larson. The Vehicle Platooning Problem: Computational Complexity and Heuristics. *Transportation Research Part C: Emerging Technologies*, 60:258–277, 2015. [see pages 39, 40, and 45]

[Mai98] Mark W Maier. Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 1(4):267–284, 1998. [see pages 76, 127, and 147]

[MBD$^+$21] Anupam Mukherjee, Partha Sarathi Barma, Joydeep Dutta, Goutam Panigrahi, Samarjit Kar, and Manoranjan Maiti. A multi-objective antlion optimizer for the ring tree problem with secondary sub-depots. *Operational Research*, pages 1–39, 2021. [see page 252]

[MCGK18] Gabriel A. Moreno, Javier Cámara, David Garlan, and Mark Klein. Uncertainty Reduction in Self-Adaptive Systems. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, pages 51–57. ACM, 2018. [see page 248]

[MCGS16] Gabriel A. Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. Efficient Decision-Making under Uncertainty for Proactive Self-Adaptation. In *2016 IEEE International Conference on Autonomic Computing*, pages 147–156, 2016. [see page 248]

[MF00] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem.

IEEE Transactions on Evolutionary Computation, 4(4):337–352, 2000. [see page 246]

[MGPR07] Riccardo Manzini, Mauro Gamberi, Alessandro Persona, and Alberto Regattieri. Design of a class based storage picker to product order picking system. *International Journal of Advanced Manufacturing Technology*, 32:811–821, 2007. [see page 254]

[MHAW16] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns. A Classification of Current Architecture-based Approaches Tackling Uncertainty in Self-Adaptive Systems with Multiple Requirements. In *Managing Trade-offs in Adaptable Software Architectures*. Elsevier, 2016. [see pages 71 and 248]

[MJMBMD17] Agustín Montero, Juan José Miranda-Bront, and Isabel Méndez-Díaz. An ILP-based local search procedure for the VRP with pickups and deliveries. *Annals of Operations Research*, 259(1):327–350, Dec 2017. [see pages 77 and 251]

[MKES21] Sumbal Malik, Manzoor Ahmed Khan, and Hesham El-Sayed. Collaborative Autonomous Driving—A Survey of Solution Approaches and Future Challenges. *Sensors*, 21(11), 2021. [see page 28]

[MKPG19] Gabriel Moreno, Cody Kinneer, Ashutosh Pandey, and David Garlan. DARTSim: An Exemplar for Evaluation and Comparison of Self-Adaptation Approaches for Smart Cyber-Physical Systems. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 181–187, 2019. [see page 2]

[MKS10] Vikas Misra, MI Khan, and UK Singh. Supply Chain Management Systems: Architecture, Design and Vision. *Journal of Strategic Innovation and Sustainability*, 6(4):96–101, 2010. [see page 49]

[MS88] Shaul Markovitch and PAUL D. SCOTT. The Role of Forgetting in Learning. In John Laird, editor, *Machine Learning Proceedings 1988*, pages 459–465. Morgan Kaufmann, San Francisco (CA), 1988. [see page 261]

[MSH08] Philipp Meisen, Thomas Seidl, and Klaus Henning. A Data-Mining Technique for the Planning and Organization of Truck

Platoons. In *Proceedings of the International Conference on Heavy Vehicles*, pages 389–402, 2008. [see pages 39, 41, and 44]

[MSS$^+$14] Vicente Milanés, Steven E. Shladover, John Spring, Christopher Nowakowski, Hiroshi Kawazoe, and Masahide Nakamura. Cooperative Adaptive Cruise Control in Real Traffic Situations. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):296–305, 2014. [see page 33]

[MST17] Christian Müller-Schloer and Sven Tomforde. *Organic Computing-Technical Systems for Survival in the Real World*. Springer, 2017. [see pages 1 and 17]

[MTZ60] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, 7(4):326–329, 1960. [see page 50]

[NFP21] Matheus Nunes, Paulo M. Fraga, and Gisele L. Pappa. Fitness Landscape Analysis of Graph Neural Network Architecture Search Spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, page 876–884, New York, NY, USA, 2021. Association for Computing Machinery. [see page 246]

[NH16] Abtin Nourmohammadzadeh and Sven Hartmann. The Fuel-Efficient Platooning of Heavy Duty Vehicles by Mathematical Programming and Genetic Algorithm. In *Proceedings of the International Conference on Theory and Practice of Natural Computing*, pages 46–57, 2016. [see pages 41 and 45]

[NH19] Abtin Nourmohammadzadeh and Sven Hartmann. Fuel-efficient truck platooning by a novel meta-heuristic inspired from ant colony optimisation. *Soft Computing*, 23(5):1439–1452, 2019. [see page 45]

[Noo15] Qais Noorshams. *Modeling and Prediction of I/O Performance in Virtualized Environments*. PhD thesis, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, February 2015. [see page 111]

[NSW$^+$12] Christoph Neumüller, Andreas Scheibenpflug, Stefan Wagner, Andreas Beham, and Michael Affenzeller. Large scale parameter meta-optimization of metaheuristic optimization algorithms with heuristiclab Hive. *Actas del VIII Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, 2012. [see pages 5 and 245]

[NZES05] P. Ngatchou, A. Zarei, and A. El-Sharkawi. Pareto Multi Objective Optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 84–91, 2005. [see pages 20 and 21]

[NZJT12] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem. *IEEE Transactions on Evolutionary Computation*, 17(5):621–639, 2012. [see page 106]

[oSCMP13] Council of Supply Chain Management Professionals. SCM Definitions and Glossary of Terms, Aug 2013. [Online; Last accessed 18. Oct. 2021]. [see pages 2 and 49]

[PA12] Erik Pitzer and Michael Affenzeller. *A Comprehensive Survey on Fitness Landscape Analysis*, pages 161–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. [see pages 246 and 262]

[Pan05] Giselher Pankratz. A Grouping Genetic Algorithm for the Pickup and Delivery Problem with Time Windows. *OR Spectrum*, 27(1):21–41, jan 2005. [see pages 77 and 251]

[PBM14] Jeremy Pitt, Dídac Busquets, and Sam Macbeth. Distributive Justice for Self-Organised Common-Pool Resource Management. *ACM Transactions on Autonomous and Adaptive Systems*, 9(3), 2014. [see page 250]

[PD21] Daniela Paddeu and Jozef Denby. Decarbonising road freight: Is truck automation and platooning an opportunity? *Clean Technologies and Environmental Policy*, pages 1–15, 2021. [see page 3]

[PDG96] Jean-Yves Potvin, Christophe Duhamel, and Francois Guertin. A Genetic Algorithm for Vehicle Routing with Backhauling. *Applied Intelligence*, 6(4):345–355, 1996. [see pages 77 and 251]

[PdSOP20] Cristiano G. Pimenta, Alex G. C. de Sá, Gabriela Ochoa, and Gisele L. Pappa. Fitness Landscape Analysis of Automated Machine Learning Search Spaces. In Luís Paquete and Christine Zarges, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 114–130, Cham, 2020. Springer International Publishing. [see page 246]

[Pet97] Charles G Petersen. An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, 17(11):1098–1111, 1997. [see pages 57, 222, and 254]

[PG18] Dmytro Pukhkaiev and Sebastian Götz. BRISE: energy-efficient benchmark reduction. In *Proceedings of the 6th International Workshop on Green and Sustainable Software*, pages 23–30, 2018. [see page 262]

[PGCP00] Martin Pelikan, David E Goldberg, and Erick Cantu-Paz. Linkage Problem, Distribution Estimation, and Bayesian Networks. *Evolutionary Computation*, 8(3):311–340, 2000. [see page 24]

[PGGM13] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013. [see page 51]

[Pig13] Tim Pigden. Missing from the Model: Orders, Drivers, Tractors and Trailers and Non-Linear Loading. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, page 1079–1084. ACM, 2013. [see pages 6, 76, and 253]

[Pin12] Michael L Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2012. [see page 250]

[PK87] Judea Pearl and Richard E Korf. Search techniques. *Annual Review of Computer Science*, 2(1):451–467, 1987. [see page 23]

[PMC$^+$12] Gilles Perrouin, Brice Morin, Franck Chauvel, Franck Fleurey, Jacques Klein, Yves Le Traon, Olivier Barais, and Jean-Marc Jézéquel. Towards flexible evolution of Dynamically Adaptive Systems. In *In Proceedings of the 34th International Conference on Software Engineering*, pages 1353–1356. IEEE, 2012. [see page 244]

[PRF16] Barry Porter and Roberto Rodrigues Filho. Losing Control: The Case for Emergent Software Systems Using Autonomous Assembly, Perception, and Learning. In *10th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 40–49. IEEE, 2016. [see pages 5 and 244]

[PRT$^+$08] Holger Prothmann, Fabian Rochner, Sven Tomforde, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. Organic Control of Traffic Lights. In Chunming Rong, Martin Gilje Jaatun, Frode Eika Sandnes, Laurence T. Yang, and Jianhua Ma, editors, *Autonomic and Trusted Computing*, pages 219–233, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. [see page 28]

[PS99] Charles G Petersen and Roger W Schmenner. An Evaluation of Routing and Volume-based Storage Policies in an Order Picking Operation. *Decision Sciences*, 30(2):481–501, 1999. [see pages 55, 56, 222, and 254]

[PSH05] Charles G Petersen, Charles Siu, and Daniel R Heiser. Improving order picking performance utilizing slotting and golden zone storage. *International Journal of Operations & Production Management*, 25(10):997–1012, 2005. [see pages 55, 56, and 254]

[PSvdWN14] Jeroen Ploeg, Dipan P. Shukla, Nathan van de Wouw, and Henk Nijmeijer. Controller Synthesis for String Stability of Vehicle Platoons. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):854–865, 4 2014. [see page 31]

[PSvN$^+$11] Jeroen Ploeg, Bart T. M. Scheepers, Ellen van Nunen, Nathan van de Wouw, and Henk Nijmeijer. Design and Experimental Evaluation of Cooperative Adaptive Cruise Control. In *Proceedings of the 14th International IEEE Intelligent Transportation Systems Conference*, pages 260–265. IEEE, 2011. [see page 32]

[Pup16] Yulia Puplaka. Implementation of platooning concept from business perspective, 2016. Student Paper. [see page 247]

[QTY10] Ali Gul Qureshi, Eiichi Taniguchi, and Tadashi Yamada. Exact solution for the vehicle routing problem with semi soft time windows and its application. *Procedia - Social and Behavioral Sciences*, 2(3):5931–5943, 2010. The Sixth International Conference on City Logistics. [see pages 77 and 251]

[Raj12] Rajesh Rajamani. *Vehicle Dynamics and Control*. Springer Science & Business Media, 2012. [see pages 30, 31, and 32]

[Rao19] Singiresu S Rao. *Engineering Optimization: Theory and Practice*. John Wiley & Sons, 2019. [see pages 19, 20, and 23]

[RCC10] Tom Robinson, Eric Chan, and Erik Coelingh. Operating Platoons On Public Motorways: An Introduction To The SARTRE Platooning Programme. In *Proceedings of the 17th World Congress on Intelligent Transport Systems*, 2010. [see pages 2, 3, 29, and 247]

[RdK01] Kees Jan Roodbergen and RenÉ de Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001. [see pages 52, 53, and 254]

[Req05] Till Requate. Dynamic incentives by environmental policy instruments–a survey. *Ecological Economics*, 54(2-3):175–195, 2005. [see page 250]

[Res02] Nicholas Rescher. *Fairness: Theory and practice of distributive justice*. Transaction Publishers, 2002. [see pages 66, 67, 81, 113, and 250]

[Ric76] John R Rice. The Algorithm Selection Problem. In *Advances in Computers*, volume 15, pages 65–118. Elsevier, 1976. [see page 5]

[RJC12] Andres J Ramirez, Adam C Jensen, and Betty HC Cheng. A taxonomy of uncertainty for dynamically adaptive systems. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 99–108. IEEE, 2012. [see pages 71 and 81]

[RKVB14] F. M. Roth, C. Krupitzer, S. Vansyckel, and C. Becker. Nature-Inspired Interference Management in Smart Peer Groups. In *International Conference on Intelligent Environments*, pages 132–139, 2014. [see page 250]

[RLR18] Jeppe Rich, Rune Larsen, and Thomas Kjær Rasmussen. Intelligent truck platooning : how to make it work. In *Proceedings of the 25th ITS World Congress*, 2018. [see pages 40 and 45]

[RMLG07] A. E. Rizzoli, R. Montemanni, E. Lucibello, and L. M. Gambardella. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1(2):135–151, sep 2007. [see pages 77 and 251]

[RMW17] Sankar Kumar Roy, Gurupada Maity, and Gerhard-Wilhelm Weber. Multi-objective two-stage grey transportation problem using utility function with goals. *Central European Journal of Operations Research*, 25(2):417–439, 2017. [see page 253]

[RMW19] Sankar Kumar Roy, Sudipta Midya, and Gerhard-Wilhelm Weber. Multi-objective multi-item fixed-charge solid transportation problem under twofold uncertainty. *Neural Computing and Applications*, 31(12):8593–8613, 2019. [see page 253]

[RMWG17] Sankar Kumar Roy, Gurupada Maity, Gerhard Wilhelm Weber, and Sirma Zeynep Alparslan Gök. Conic scalarization approach to solve multi-choice multi-objective transportation problem with interval goal. *Annals of Operations Research*, 253(1):599–620, 2017. [see page 253]

[RR83] H Donald Ratliff and Arnon S Rosenthal. Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. *Operations Research*, 31(3):507–521, 1983. [see pages 57, 58, and 254]

[RRFS07] Matthias Rockl, Patrick Robertson, Korbinian Frank, and Thomas Strang. An architecture for situation-aware driver assistance systems. In *2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring*, pages 2555–2559. IEEE, 2007. [see pages 5 and 243]

[RSCMT19] J Reyes, E Solano-Charris, and J Montoya-Torres. The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations*, 10(2):199–224, 2019. [see page 74]

[RSM20] Bochra Rabbouch, Foued Saâdaoui, and Rafaa Mraihi. Empirical-type simulated annealing for solving the capacitated vehicle routing problem. *Journal of Experimental & Theoretical Artificial Intelligence*, 32(3):437–452, 2020. [see pages 77 and 251]

[RTLZ00] Rajesh Rajamani, Han-Shue Tan, Boon Kait Law, and Wei-Bin Zhang. Demonstration of Integrated Longitudinal and Lateral Control for the Operation of Automated Vehicles in Platoons. *IEEE Transactions on Control Systems Technology*, 8(4):695–708, 2000. [see page 31]

[SAAS14] Maram Shqair, Safwan Altarazi, and Sameh Al-Shihabi. A statistical study employing agent-based modeling to estimate the effects of different warehouse parameters on the distance traveled in warehouses. *Simulation Modelling Practice and Theory*, 49:122–135, 2014. [see page 254]

[SBH17] Dietrich Steinmetz, Gerrit Burmester, and Sven Hartmann. A Fast Heuristic for Finding Near-Optimal Groups for Vehicle Platooning in Road Networks. In *Proceedings of the International Conference on Database and Expert Systems Applications*, pages 395–405. Springer International Publishing, 2017. [see page 41]

[Seg16] Michele Segata. *Safe and Efficient Communication Protocols for Platooning Control*. Phd thesis, University of Innsbruck, 2 2016. [see pages 29, 67, and 263]

[SGD11] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, 2011. [see page 46]

[SGPB17] Sanny Schmid, Ilias Gerostathopoulos, Christian Prehofer, and Thomas Bures. Self-Adaptation Based on Big Data Analytics: A Model Problem and Tool. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 102–108, 2017. [see page 249]

[SHKM14] Yuan Sun, Saman K. Halgamuge, Michael Kirley, and Mario A. Munoz. On the selection of fitness landscape analysis metrics for continuous optimization problems. In *7th International Conference on Information and Automation for Sustainability*, pages 1–6, 2014. [see page 246]

[SIS13] Stefan Solyom, Arash Idelchi, and Badr Bin Salamah. Lateral Control of Vehicle Platoons. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 4561–4565. IEEE, 10 2013. [see page 33]

[SJB$^+$14] Michele Segata, Stefan Joerer, Bastian Bloessl, Christoph Sommer, Falko Dressler, and Renato Lo Cigno. PLEXE: A Platooning Extension for Veins. In *Proc. VNC*, pages 53–60, 2014. [see pages 46 and 263]

[SKG12] Natanaree Sooksaksun, Voratas Kachitvichyanukul, and Dah-Chuan Gong. A class-based storage warehouse design using a particle swarm optimisation algorithm. *International Journal of Operational Research*, 13(2):219–237, 2012. [see page 254]

[Ski11] Rafał Skinderowicz. Co-operative, Parallel Simulated Annealing for the VRPTW. In *Computational Collective Intelligence. Technologies and Applications*, pages 495–504. Springer Berlin Heidelberg, 2011. [see pages 77 and 251]

[SKSB21] Timo Sturm, Christian Krupitzer, Michele Segata, and Christian Becker. A Taxonomy of Optimization Factors for Platooning. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6097–6114, 2021. [see pages 3, 29, and 37]

[SKVPA16] Elham Semsar-Kazerooni, Jan Verhaegh, Jeroen Ploeg, and Mohsen Alirezaei. Cooperative adaptive cruise control: An artificial potential field approach. In *IEEE Intelligent Vehicles Symposium*, pages 361–367, Göteborg, Sweden, 6 2016. IEEE. [see page 33]

[SLB19] Xudong Sun, Jiali Lin, and Bernd Bischl. ReinBo: Machine Learning pipeline search and configuration with Bayesian Optimization embedded Reinforcement Learning. *arXiv preprint arXiv:1904.05381*, 2019. [see page 245]

[SLHB21] Yong-Jun Shin, Lingjun Liu, Sangwon Hyun, and Doo-Hwan Bae. Platooning LEGOs: An Open Physical Exemplar for Engineering Self-Adaptive Cyber-Physical Systems-of-Systems. In

2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pages 231–237, 2021. [see page 2]

[SLM+17] Vadim Sokolov, Jeffrey Larson, Todd Munson, Josh Auld, and Dominik Karbowski. Platoon formation maximization through centralized routing and departure time coordination. Technical report, Argonne National Laboratory, 2017. [see page 41]

[SLMS20] Sergey Sukhov, Mikhail Leontev, Alexander Miheev, and Kirill Sviatov. Prevention of catastrophic interference and imposing active forgetting with generative methods. *Neurocomputing*, 400:73–85, 2020. [see page 261]

[SM09] Kate A Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):1–25, 2009. [see pages 5, 108, and 244]

[SMM08] Alexander Scheidler, Daniel Merkle, and Martin Middendorf. Congestion Control in Ant Like Moving Agent Systems. In *Proceedings of Biologically-Inspired Collaborative Computing*, pages 33–43. Springer, 2008. [see page 250]

[SR16] Ali Mohammad Shahdaei and Amir Masoud Rahimi. Solving vehicule routing problem with simultaneous pick-up and delivery with the application of genetic algirithm. In *Indian Journal of Fundamental and Applied Life Sciences*, volume 6, pages 247–259, 2016. [see pages 77 and 251]

[SSV+15] Stefania Santini, Allesandro Salvi, Antonio Saverio Valente, Antonio Pescap, Michele Segata, and Renato Lo Cigno. A Consensus-based Approach for Platooning with Inter-Vehicular Communications. In *Proceedings of the 2015 IEEE Conference on Computer Communications*, pages 1158–1166, 2015. [see page 33]

[SSV+19] Stefania Santini, Alessandro Salvi, Antonio Saverio Valente, Antonio Pescapè, Michele Segata, and Renato Lo Cigno. Platooning Maneuvers in Vehicular Networks: a Distributed and Consensus-Based Approach. *IEEE Transactions on Intelligent Vehicles (T-IV)*, 4(1):59–72, 3 2019. [see page 33]

[Sus08] Joseph S Sussman. *Perspectives on intelligent transportation systems (ITS)*. Springer Science & Business Media, 2008. [see pages 2, 27, and 28]

[SWH11] W.Y. Szeto, Yongzhong Wu, and Sin C. Ho. An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, 215(1):126–135, 2011. [see pages 77 and 251]

[SWM19] Stepan Shevtsov, Danny Weyns, and Martina Maggio. SimCA*: A Control-Theoretic Approach to Handle Uncertainty in Self-Adaptive Systems with Guarantees. *ACM Transactions on Autonomous and Adaptive Systems*, 13(4), 2019. [see pages 248 and 249]

[TBJ17] Valerio Turri, Bart Besselink, and Karl H. Johansson. Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning. *IEEE Transactions on Control Systems Technology*, 25(1):12–28, 2017. [see page 33]

[TG10] A. Serdar Tasan and Mitsuo Gen. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. In *Computers & Industrial Engineering*, volume 62. IEEE, 2010. [see pages 77 and 251]

[TGBM17] Erfan Babaee Tirkolaee, Alireza Goli, Mani Bakhsi, and Iraj Mahdavi. A robust multi-trip vehicle routing problem of perishable products with intermediate depots and time windows. *Numerical Algebra, Control & Optimization*, 7(4):417–433, 2017. [see page 252]

[THHLB13] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013. [see page 245]

[THS$^+$14] S. Tomforde, J. Hähner, H. Seebach, W. Reif, B. Sick, A. Wacker, and I. Scholtes. Engineering and Mastering Interwoven Systems. In *Workshop Proceedings on Architecture of Computing Systems*, 2014. [see page 249]

[THS$^+$18] Erfan Babaee Tirkolaee, Ali Asghar Rahmani Hosseinabadi, Mehdi Soltani, Arun Kumar Sangaiah, and Jin Wang. A Hybrid Genetic Algorithm for Multi-Trip Green Capacitated Arc Routing Problem in the Scope of Urban Services. *Sustainability*, 10(5), 2018. [see page 252]

[THWM20] Erfan Babaee Tirkolaee, Shaghayegh Hadian, Gerhard-Wilhelm Weber, and Iraj Mahdavi. A robust green traffic-based routing problem for perishable products distribution. *Computational Intelligence*, 36(1):80–101, 2020. [see page 253]

[TJS16] Sadayuki Tsugawa, Sabina Jeschke, and Steven E Shladover. A review of truck platooning projects for energy savings. *IEEE Transactions on Intelligent Vehicles*, 1(1):68–77, 2016. [see page 29]

[TPB$^+$11] Sven Tomforde, Holger Prothmann, Jürgen Branke, Jörg Hähner, Moez Mnif, Christian Müller-Schloer, Urban Richter, and Hartmut Schmeck. Observation and Control of Organic Systems. In *Organic Computing–A Paradigm Shift for Complex Systems*, pages 325–338. Springer, 2011. [see pages 2, 17, and 88]

[TPC08] Jorge Tavares, Francisco B. Pereira, and Ernesto Costa. Multidimensional Knapsack Problem: A Fitness Landscape Analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(3):604–616, 2008. [see page 246]

[TV14] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. Society for Industrial and Applied Mathematics, 2014. [see page 51]

[TWH01] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 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, 2001. [see page 106]

[TYA$^+$17] M. Noumbissi Tchoupo, A. Yalaoui, L. Amodeo, F. Yalaoui, and F. Lutz. Ant Colony Optimization Algorithm for Pickup and Delivery Problem with Time Windows. In Antonio Sforza and Claudio Sterle, editors, *Springer Proceedings in Mathematics & Statistics*, pages 181–191. Springer International Publishing, 2017. [see pages 77 and 251]

[UAB$^+$08] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt Mcnaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William Red Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, and Michael Taylor. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. [see page 31]

[Vau99] T.S. Vaughan. The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, 37(4):881–897, 1999. [see page 254]

[VCG$^+$12] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, 60(3):611–624, 2012. [see pages 77 and 251]

[VCIC15] Lucian Vinţan, Radu Chiş, Muhammad Ali Ismail, and Cristian Coţofană. Improving Computing Systems Automatic Multiobjective Optimization Through Meta-Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(7):1125–1129, 2015. [see pages 5 and 245]

[vGRCdK18] Teun van Gils, Katrien Ramaekers, An Caris, and René B. M. de Koster. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, 267:1–15, 05 2018. [see pages 6, 75, and 254]

[VJD15] Sebastian Van De Hoef, Karl H Johansson, and Dimos V Dimarogonas. Fuel-Optimal Centralized Coordination of Truck Platooning Based on Shortest Paths. In *Proceedings of the 2015 American Control Conference*, pages 3740–3745, 2015. [see pages 38, 39, 40, 41, and 45]

[VJD18] Sebastian Van De Hoef, Karl Henrik Johansson, and Dimos V Dimarogonas. Fuel-Efficient En Route Formation of Truck Platoons. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):102–112, 2018. [see pages 39 and 42]

[VMDJ19] Sebastian Van De Hoef, Jonas Mårtensson, Dimos V. Dimarogonas, and Karl Henrik Johansson. A Predictive Framework for Dynamic Heavy-Duty Vehicle Platoon Coordination. *ACM Transactions on Cyber-Physical Systems*, 4(1), 2019. [see pages 41 and 42]

[Voi31] Bernhard Friedrich Voigt. Der Handlungsreisende, wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein. *Commis-Voageur, Ilmenau*, 1831. [see page 50]

[VWHK13a] Willem Van Willigen, Evert Haasdijk, and Leon Kester. A multi-objective approach to evolving platooning strategies in intelligent transportation systems. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 1397–1404. Association for Computing Machinery, 2013. [see page 40]

[VWHK13b] Willem Van Willigen, Evert Haasdijk, and Leon Kester. Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives. In *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 1448–1455, 2013. [see page 44]

[VWMA11] Pieter Vromant, Danny Weyns, Sam Malek, and Jesper Andersson. On Interacting Control Loops in Self-Adaptive Systems. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '11, page 202–207, New York, NY, USA, 2011. Association for Computing Machinery. [see page 28]

[WAY$^+$16] Shuai Wang, Shaukat Ali, Tao Yue, Yan Li, and Marius Liaaen. A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering. In *Proceedings of the 38th International Conference on Software Engineering*, pages 631–642. ACM, 2016. [see pages 21 and 219]

[WC12] Hsiao-Fan Wang and Ying-Yen Chen. A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, 62(1):84–95, 2012. [see pages 77 and 251]

[WLZY18] Mang Wang, Bin Li, Guofu Zhang, and Xin Yao. Population Evolvability: Dynamic Fitness Landscape Analysis for Population-Based Metaheuristic Algorithms. *IEEE Transactions on Evolutionary Computation*, 22(4):550–563, 2018. [see page 246]

[WM97] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. [see pages 4, 73, 74, 75, 81, 107, 110, 113, and 119]

[WS03] Anne Wade and Said Salhi. An Ant System Algorithm for the Mixed Vehicle Routing Problem with Backhauls. In *Metaheuristics: Computer Decision-Making*, pages 699–719. Springer US, 2003. [see pages 77 and 251]

[WTKD04] William E. Walsh, Gerald Tesauro, Jeffrey O. Kephart, and Rajarshi Das. Utility Functions in Autonomic Systems. In *International Conference on Autonomic Computing*, pages 70–77, 2004. [see pages 122 and 249]

[WW02] Jane Webster and Richard T Watson. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, 26(2):xiii–xxiii, 2002. [see page 34]

[WZW$^+$16] Jiahai Wang, Ying Zhou, Yong Wang, Jun Zhang, C. L. Philip Chen, and Zibin Zheng. Multiobjective Vehicle Routing Problems With Simultaneous Delivery and Pickup and Time Windows: Formulation, Instances, and Algorithms. *IEEE Transactions on Cybernetics*, 46(3):582–594, 2016. [see pages 77 and 251]

[WZZL18] Lijun Wei, Zhenzhen Zhang, Defu Zhang, and Stephen C.H. Leung. A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 265(3):843–859, 2018. [see pages 77 and 251]

[XGN$^+$10] Bo Xing, Wen-Jing Gao, Fulufhelo V Nelwamondo, Kimberly Battle, and Tshilidzi Marwala. Ant colony optimization for automated storage and retrieval system. In *IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2010. [see page 254]

[XPN16] Haitao Xing, Jeroen Ploeg, and Henk Nijmeijer. Padé Approximation of Delays in Cooperative ACC Based on String Stability Requirements. *IEEE Transactions on Intelligent Vehicles (T-IV)*, 1(3):277–286, 9 2016. [see page 31]

[XSBC14] Xiao-Feng Xie, Stephen F Smith, Gregory J Barlow, and Ting-Wei Chen. Coping with real-world challenges in real-time urban traffic control. In *Compendium of Papers of the 93rd Annual Meeting of the Transportation Research Board*, pages 1–15, 2014. [see page 2]

[XSJ21] Xi Xiong, Junyi Sha, and Li Jin. Optimizing Coordinated Vehicle Platooning: An Analytical Approach Based on Stochastic Dynamic Programming. *Transportation Research Part B: Methodological*, 150:482–502, 2021. [see page 45]

[YMK$^+$18] Mohamed El Yafrani, Marcella S. R. Martins, Mehdi El Krari, Markus Wagner, Myriam R. B. S. Delgado, Belaïd Ahiod, and Ricardo Lüders. A Fitness Landscape Analysis of the Travelling Thief Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, page 277–284, New York, NY, USA, 2018. Association for Computing Machinery. [see page 246]

[YYY09] Bin Yu, Zhong-Zhen Yang, and Baozhen Yao. An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196(1):171–176, 2009. [see pages 77 and 251]

[ZBL$^+$19] Marwin Züfle, André Bauer, Veronika Lesch, Christian Krupitzer, Nikolas Herbst, Samuel Kounev, and Valentin Curtef. Autonomic Forecasting Method Selection: Examination and Ways Ahead. In *Proceedings of the 16th IEEE International Conference on Autonomic Computing*. IEEE, June 2019. [see page xxv]

[ZC18] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly, 2018. [see page 123]

[ZHO$^+$18] Yuanyuan Zhang, Mark Harman, Gabriela Ochoa, Guenther Ruhe, and Sjaak Brinkkemper. An Empirical Study of Meta- and Hyper-Heuristic Search for Multi-Objective Release Planning. *ACM Transactions on Software Engineering and Methodology*, 27(03), 2018. [see pages 5 and 245]

[ZMJ16] Wei Zhang, Xiaoliang Ma, and Erik Jenelius. Planning of heavy-duty vehicle platoon formulation: basic scheduling problem considering travel time variance. In *Proceedings of the Transportation Research Board 95th Annual Meeting*, 2016. [see page 41]

[ZML$^+$21] Marwin Züfle, Felix Moog, Veronika Lesch, Christian Krupitzer, and Samuel Kounev. A machine learning-based workflow for automatic detection of anomalies in machine tools. *ISA Transactions*, 2021. Impact Factor (2020): 5.468. [see page xxiii]

[ZSK17] Wei Zhang, Marcus Sundberg, and Anders Karlström. Platoon coordination with time windows: an operational perspective. *Transportation Research Procedia*, 27:357–364, 2017. [see page 44]

[ZYS$^+$16] Xinglin Zhang, Zheng Yang, Wei Sun, Yunhao Liu, Shaohua Tang, Kai Xing, and Xufei Mao. Incentives for Mobile Crowd Sensing: A Survey. *IEEE Communications Surveys & Tutorials*, 18(1):54–67, 2016. [see page 250]