JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche
Fakultät

Adaptive Heuristic Approaches for Dynamic Vehicle Routing - Algorithmic and Practical Aspects

DISSERTATION

zur Erlangung des akademischen Grades

Doktor

im Doktoratsstudium der

Technischen Wissenschaften

Eingereicht von:
Stefan Vonolfen, MSc

Angefertigt am:
Institut für Formale Modelle und Verifikation

Beurteilung:
Priv.-Doz. Dr. Michael Affenzeller (Betreuung)
Univ.-Prof. Dr. Karl Dörner

Linz, Juni, 2014

Acknowledgments

The work presented in this thesis would not have been possible without the many fruitful discussions with my colleagues from the research group *Heuristic and Evolutionary Algorithms Laboratory* (HEAL) and without the HeuristicLab optimization environment as a software infrastructure (the web pages of all members of HEAL as well as further information about HeuristicLab can be found at: `http://www.heuristiclab.com`).

Particularly, I would like to thank Michael Affenzeller for his guidance concerning the algorithmic aspects of this thesis as my supervisor as well as providing a very supportive working environment as the research group head. I would also like to thank Stefan Wagner for triggering my research interest in metaheuristic algorithms during the supervision of my bachelor and master thesis. The discussions with my colleagues Andreas Beham and Erik Pitzer about vehicle routing, fitness landscape analysis, and algorithm selection led to many ideas presented in this thesis. Michael Kommenda provided important insights on genetic programming and Monika Kofler contributed knowledge about storage assignment. Stephan Hutterer was working on the generation of policies for smart grids and pointed out many links to related literature.

I would also like to thank Prof. Karl Dörner from the institute for production and logistics management at the JKU for giving me the possibility to present my work at the ORP3 workshop as well as during a seminar at his institute. The workshop provided the possibility to submit my work to a renowned international operations research journal (the article is currently in third revision and is based on topics presented in this thesis). The discussions at the seminar and the workshop as well as the review comments on the journal article provided many valuable insights about algorithmic extensions and modeling aspects.

Last, but not least I would also like to acknowledge the received funding. The work described in this thesis was done within the Josef Ressel Centre for Heuristic Optimization supported by the Austrian Research Promotion Agency (FFG) as well as the Regio 13 program sponsored by the European Regional Development Fund and by Upper Austrian public funds.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Dissertation ist mit dem elektronisch übermittelten Textdokument identisch.

Zusammenfassung

Die dynamische Tourenplanung gewinnt im Bereich der Logistikoptimierung mehr und mehr an Bedeutung. Die Entwicklung von immer effizienteren Optimierungalgorithmen sowie technologische Verbesserungen von Telematiksystemen ermöglichen den Einsatz von praxisnahen Modellen in Unternehmen.

Bei der Entwicklung von Optimierungsverfahren wird allerdings oftmals ein Kompromiss zwischen breiter Anwendbarkeit und Spezialisierung gemacht. Hochspezialisierte Lösungsstrategien funktionieren in gewissen Situationen zwar besser, machen jedoch Abstriche im Hinblick auf Robustheit. Aus dieser Beobachtung heraus ergibt sich die in dieser Arbeit verfolgte Vision eines adaptiven Entscheidungs-Unterstützungssystems für dynamische Tourenplanungsumgebungen mit sich ändernden Problemcharakteristiken.

Eine laufende Anpassung der Lösungsstrategien erfordert die Verlagerung der Algorithmenentwicklung auf eine höhere Abstraktionsebene. Eine Meta-Betrachtungsweise von Algorithmen erlaubt eine semi-automatische Generierung von spezialisierten Lösungsstrategien als auch eine adaptive Algorithmenauswahl auf Basis von Problemcharakteristiken. Auf diese Weise werden die Stärken von verschiedenen spezialisierten Algorithmen in sich ändernden Problemumgebungen kombiniert.

Aufbauend auf diesen Konzepten wird ein algorithmisches Rahmenwerk vorgestellt, welches aus drei grundlegenden Bausteinen besteht. Die simulationsbasierte Optimierung erlaubt die Erstellung von praxisnahen Modellen mit stochastischen Einflussgrößen und komplexen dynamischen Interaktionen. Auf Basis dieser Modelle werden unter Anwendung von Evolutionären Algorithmen und bestärkendem Lernen spezialisierte Lösungsstrategien gesucht. Diese werden zu Algorithmenportfolios kombiniert, welche eine situative Auswahl anhand der Problemcharakteristiken ermöglichen.

Die bedeutendste Leistung dieser Arbeit ist die semi-automatische Generierung und Adaption von algorithmischen Strategien, was durch die Verlagerung der Algorithmenentwicklung auf eine höhere Abstraktionsebene erreicht wird. Zukünftige Entwicklungen, wie die Integration von maschinellem Lernen oder die Erforschung der Lösungsraumcharakteristiken, stehen vor allem im Kontext von autonom agierenden adaptiven Tourenplanungssystemen. Das vorgestellte algorithmische Rahmenwerk bietet hierfür eine Grundlage.

Abstract

Dynamic vehicle routing is an active field of research due to the practical relevance as well as the advances in operations research and telematics. Advanced algorithmic approaches are being developed and at the same time, more and more realistic problem formulations are being investigated enabling to transfer the findings into practice.

The main vision pursued in this thesis is a decision support system for dynamic vehicle routing problems that is adaptive in terms of problem characteristics and automatically changes its algorithmic strategies based on the environment. The motivation for such a system stems from the fact that there is a tradeoff between generalization and specialization in algorithm design. On the one hand, research on algorithms for dynamic vehicle routing problems focused mainly on robust behavior over a large range of problem instances. On the other hand, it has been identified that highly specialized policies have the potential to outperform these general strategies while non-robust behavior was observed for them as a trade-off.

The methodological developments presented in this thesis aim to solve this dilemma by raising the abstraction level of algorithm design for dynamic vehicle routing problems to a meta-level to pursue the goal of self-adaptive algorithmic strategies. On the meta-level a semi-automatic generation as well as an adaptive selection of policies based on the problem characteristics is performed combining the strengths of several specialized policies in changing environments.

Three essential building blocks of an adaptive algorithmic framework for dynamic vehicle routing are identified. Simulation optimization allows modeling practical variants with rich side constraints. Three practical case-studies from production and logistics are investigated while highlighting the transfer of findings into practice. Based on a simulation model, specialized routing policies are generated by means of direct policy search and reinforcement learning. Routing policies for three different variants are evolved. Human-designed as well as generated routing policies are combined to an algorithm portfolio allowing a dynamic situational policy selection. A case-study is presented illustrating the potential of the methodology.

The main achievement of this thesis is raising the abstraction level for algorithm design in the context of dynamic vehicle routing by means of the proposed algorithmic framework. While the semi-automated adaption of the algorithmic strategies has been reached, future research should focus on a fully autonomous system that learns in a changing environment. Such a system requires the incorporation of machine learning and a fundamental understanding of problem characteristics linked to algorithm performance.

Contents

Chapter 1

Introduction

The vehicle routing problem (VRP) is an important problem class in operations research (OR) because it can be used to model various types of transportation problems. Since its original formulation by Dantzig and Ramser (1959) many variants have emerged and have been successfully applied in practice (Golden et al., 2008, Laporte, 2009).

In many markets, customers are demanding a flexible and timely fulfillment of their requests and transport logistics companies are faced with increasingly competitive environments with global players. At the same time, more and more real-time information is available during the planning process. To address these challenges, advances in the fields of telematics and operations research have enabled the application of rich and dynamic VRP variants.

1.1 Motivation and Research Questions

Challenges in contemporary vehicle routing research are rich models that include many practical side constraints (Hartl et al., 2006) as well as dynamic and stochastic information (Pillac et al., 2012a).

There is a growing body of literature on dynamic variants of the vehicle routing problem while it still remains a challenging combinatorial optimization problem and is a subject of active research. These developments have led to many different variants since the seminal work of Psaraftis (1988). Successful practical applications include the distribution of heating oil, taxi cab services, on-demand transportation of elderly people or courier services (Pillac et al., 2012a).

The algorithmic developments proposed in this thesis have been triggered by research directions given in surveys on dynamic vehicle routing such as

Ichoua et al. (2007), Larsen et al. (2008), and Pillac et al. (2012a) while the main inspiration were the statements of Larsen et al. (2008) which are still highly relevant in the context of recent developments.

As pointed out by Larsen et al. (2008), while surveying future research directions, they

> [...] believe a new generation of DVRP algorithms will blend the effectiveness of advanced methods, tailored to **take advantage of special problem structures and advanced knowledge**, with the efficiency of parallel implementations, and the ever growing computing power of workstations to solve increasingly larger and more realistic problems.

In this statement the need for specialized approaches is highlighted that take advantage of the problem structure and knowledge about the characteristics of the problem environment. The need for heuristics tailored to the problem characteristics corresponds to the well-known *no free lunch theorem* (Wolpert and Macready, 1997). The development and parametrization of specialized heuristics for different variants of dynamic VRPs has been mainly done manually in the past while it remains a research challenge to algorithmically generate specialized policies. In this context, the modeling and optimization of more realistic problem formulations is an important research topic.

Detailing their vision of a new generation of dynamic VRP (DVRP) algorithms, Larsen et al. (2008) state the following:

> The overnight courier mail service provider environment represents a good model for the use of such new **hybrid approaches**. The morning subproblem is often weakly dynamic while the afternoon one is moderately dynamic. Therefore, a reoptimization algorithm could first plan a set of morning delivery routes. In case of urgent call-in requests, the algorithm could insert the new requests into the predetermined delivery routes. In turn, the afternoon pickup problem, would use fast algorithms for online routing that would take advantage of a priori information on future requests.

Most existing work in the field of dynamic VRPs considers environments where the characteristics such as spatial and temporal properties of the appearing requests do not change over time. As a result, solution methods have been developed that are parameterized and tailored to certain characteristics. An extensive methodological framework for dealing with changing environments is not available.

1.2 Synopsis

The main research goal of this thesis is to propose adaptive heuristic approaches for dynamic vehicle routing problems with special emphasis on practical applicability. The main vision is a decision support system that learns from previous observations and constantly adapts its algorithmic strategies to changing problem characteristics.

Several contributions are made to reach this goal based on current research-challenges in the area of dynamic vehicle routing. In particular, this work makes three contributions beyond the current state-of-the-art by proposing a modeling approach for the **simulation-based optimization of production and logistic scenarios** as well as algorithmic frameworks for the **generation of specialized routing policies** and a **situational selection of solution techniques**. The developed techniques are essential building blocks for adaptive decision support systems. The three main innovative aspects of this thesis are highlighted in the following.

Firstly, a generic simulation and optimization environment that can be adapted to model rich practical variants is presented. To ensure the practical applicability, case-studies are performed using real-world data and the results are validated together with domain experts and transfered into practice. Concerning the modeling of more realistic problem formulations in production and logistics, the main contributions are:

- A generic modeling, simulation and optimization environment for different variants of real-world dynamic vehicle routing problems is presented. Two concrete realizations for pickup and delivery and inventory routing problems are outlined.

- Based on the generic simulation optimization model, three practical case-studies are presented that are based on real-world data and have been validated together with domain experts to demonstrate the practical applicability of the modeling as well as algorithmic approaches:

  An optimization of transport activities in the cold-charge steel production process is presented and bottlenecks are analyzed (based on Vonolfen et al. (2013b)).

  An integrated optimization of warehousing and transport activities in the production of firefighting vehicles has been considered analyzing the interrelations between the sub-activities (based on Vonolfen et al. (2012b)).

  A vendor-managed inventory for the distribution of groceries is investigated (based on Vonolfen et al. (2013a)). The study considers

mixed scenarios where only a part of the customers is switched to a vendor managed inventory to perform a sensitivity analysis in terms of endogenous and exogenous influence factors.

Secondly, an algorithmic framework is proposed for semi-automatically generating specialized policies that are adapted to the problem environment. The main contributions are:

- Proposal of a framework for the algorithmic generation of specialized policies based on a black-box simulation model by means of evolutionary direct policy search and reinforcement learning. The applicability of this framework is shown for several variants of dynamic vehicle routing problems.

- Evolution of replenishment rules for inventory routing problems (based on Vonolfen et al. (2013a)). The evolved rules are utilized for the evaluation of various strategic scenarios in the context of a sensitivity analysis based on real-world data from a company dealing with multi-channel retailing.

- Generation of dispatching rules for dial-a-ride problems within an agent environment (based on Vonolfen et al. (2013c)). The performance of the evolved rules is compared with a planning algorithm in terms of solution quality and runtime on a set of test instances. It is shown, that the rules are suitable for highly dynamic environments where planning ahead might not be feasible.

- Automatic evolution of waiting strategies for pickup and delivery problems based on historical data. The generated waiting heuristics outperform existing approaches on a set of standard benchmark instances. The influence of spatial and temporal problem properties is investigated to evaluate the potential of anticipatory waiting.

Thirdly, the algorithmic approach proposed in this work to deal with changing problem environments is to combine several specialized heuristics and situationally select an appropriate solution method. The contributions in the context of this emerging research direction are:

- Proposal of a generic methodological framework for algorithmic selection of solution techniques based on a portfolio of specialized heuristics with different strengths and weaknesses. Resulting from the methodology, several research questions arrive such as problem feature extraction and algorithm selection.

- Discussion of possible approaches for problem feature extraction which include fitness landscape analysis or the usage of problem-specific features.
- Algorithm selection and performance prediction based on knowledge about previous performance. Links are presented to existing literature dealing with algorithm selection.
- Application of the method to practically relevant case-studies. Several possible applications to scenarios with changing characteristics, such as data quality, request intensity, or seasonal fluctuations, are given. The methodology is applied to a scenario with changing data quality where several solution methods are combined.

The three research topics are closely connected to each other in the context of a self-adapting decision support system. The situational selection builds on specialized algorithms that are generated automatically. The algorithmic approaches rely on a realistic simulation and optimization model of the system. The general aim is, that these algorithmic developments will lead to decision support systems that learn and adapt within changing problem environments by utilizing knowledge about the problem structure and the implications on the solution quality.

1.3 Chapter Overview

In Chapter 2, the theoretical foundations and a literature survey are provided linking the findings to related work. Chapter 3 deals with the simulation-based optimization approach to model practical variants. The simulation of data is detailed as well as the optimization framework. Three practical case-studies from production and logistics are presented. Chapter 4 and Chapter 5 outline the methodological aspects of the two algorithmic research topics. In Chapter 4, a framework for automatically generating specialized policies that are adapted to the problem environment is proposed. Dealing with changing problem characteristics during the planning process by situational selection of a suitable solution method is the main topic of Chapter 5. Computational results are presented in Chapter 6. The main achievements are summarized in Chapter 7 and an outlook is given on future research directions.

Chapter 2

Dynamic Vehicle Routing

This chapter provides theoretical, methodological and practical foundations of the field of dynamic vehicle routing. Section 2.1 gives a brief overview of the extensive research that has been conducted in the context of vehicle routing problems, provides basic models and theoretical as well as practical considerations about dynamic variants. Section 2.2 surveys solution methods and highlights important success factors such as the consideration of future requests or parallelization. In Section 2.3 the technical requirements are discussed for implementing a decision support system for dynamic vehicle routing in practice. In Section 2.4 current research directions are surveyed.

2.1 Foundations

The dynamic vehicle routing problem has some important aspects compared to the static variant that impose additional challenges and require specialized algorithmic strategies. In the following, the archetypical variant of the static vehicle routing problem will be introduced and the extensive work conducted in this area will be outlined. The step from static to dynamic vehicle routing is detailed as well as the main characteristics of dynamic variants and the resulting challenges.

2.1.1 The Vehicle Routing Problem

The first paper on vehicle routing was published by Dantzig and Ramser (1959) and since then there have been over 50 years of extensive research and constantly growing literature (Laporte, 2009). Numerous practical applications resulted and a tool industry has emerged - a recent survey of commercial vehicle routing software lists 15 different vendors (Hall, 2012).

Books and survey articles within the extensive scientific literature, to name but a few, include Toth and Vigo (2002), Golden et al. (2008), Gendreau et al. (2008), Laporte (2009), and Baldacci et al. (2012).

The vehicle routing problem (VRP) is not a single problem but rather a class of problems since in many practical applications variants are required that have different operational constraints and objectives (cf. Laporte (2009)). This stems from the diverse application areas, such as the distribution of goods or transportation of people, and operating rules, such as capacity constraints or time windows. In general, the VRP can be defined as the design of a set of routes for a fleet of vehicles to service geographically scattered customer demands while considering operational constraints and objectives (cf. Toth and Vigo (2002)).

Basic Models

As pointed out by Laporte (2009), methodological developments usually work with archetypal versions of the problem. The most basic variant is the capacitated vehicle routing problem (CVRP) which is also denoted as the classical VRP. The CVRP is known to be NP-hard and generalizes the traveling salesman problem (TSP). Even the most basic variant is thus generally considered as an intractable problem.

The graph theoretic notation of the CVRP used in this work follows Toth and Vigo (2002): The complete graph $G = (V, A)$ is defined by the vertex set $V = \{0, ..., n\}$ and the arc set $A = \{(i, j) : i, j \in V, i \neq j\}$. Vertex $i = 0$ corresponds to the depot and vertices $i = 1, ..., n$ correspond to the customers. For each arc $(i, j) \in A$ a nonnegative value is given which corresponds to the travel cost from vertex i to j. The customers ($i = 1, ..., n$) are each associated with a nonnegative demand d_i and are served by a fleet of K homogeneous vehicles with capacity C. Each vehicle performs at most one route starting and ending at the depot that does not exceed the capacity restrictions and each customer is served by exactly one vehicle. The aim is to minimize the total travel costs while satisfying all demands.

There are several mathematical programming formulations that are used to model the CVRP. Here a short overview of the main modeling approaches is given, for details the reader is referred to Toth and Vigo (2002) which is the basis for this brief summary. The three main approaches are *vehicle flow*, *commodity flow* and *set partitioning* formulations.

In models using *vehicle flow* formulations the focus is on transitions of vehicles between customers. Models of this type are thus especially suitable when the main constraints concern the traversal of arcs. The problem model

stems from the work of Laporte and Nobert (1983) and can be stated as[1]:

$$\textit{minimize} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.1}$$

$$\textit{subject to} \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{2.2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{2.3}$$

$$\sum_{i \in V} x_{i0} = K, \tag{2.4}$$

$$\sum_{j \in V} x_{0j} = K, \tag{2.5}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{2.7}$$

The model stated here is a two-index vehicle flow formulation. The binary decision variable x_{ij} states if a vehicle traverses a certain arc and takes the value $x_{ij} = 1$ if the arc belongs to the optimal solution and $x_{ij} = 0$ otherwise. Three index flow formulations also contain an index for the vehicle that traverses the arc while the model presented here assumes that all vehicles are homogeneous and it is not significant what vehicle is assigned to which route. The variable c_{ij} gives the costs for traversing arc (i, j). The objective is minimizing the sum of all traversed arcs while following the constraints (2.1). The indegree constraint (2.2) makes sure that exactly one arc enters while the outdegree constraint (2.3) ensures that one arc leaves each vertex. The depot vertex is entered and left by exactly K arcs which is the number of vehicles (2.4, 2.5). The connectivity of the solution as well as the vehicle capacity restrictions are ensured by the capacity-cut constraints (2.6). In that definition, $r(s)$ is defined as the minimum number of vehicles needed to serve a nonempty subset of all customers(S). The number of vertices entering each subset must be greater or equal the capacity requirements. This ensures subtour elimination as well as the capacity limitations.

Models using *commodity flow* formulations focus on the flow of commodities along the arcs traversed by the vehicles and were first introduced by Garvin et al. (1957). For each arc, a vehicle load carried on that arc is defined. These formulations can be used to model additional constraints compared to vehicle flow models such as vehicle assignment to a route.

[1] Toth and Vigo (2002)

Models formulated as *set partitioning* problems view a VRP solution as a set of feasible routes. Instead of selecting arcs, as in the other two formulations, the routes are selected that are part of the final solution. This formulation was first provided by Balinski and Quandt (1964). As pointed out by Laporte (2009) a direct application of this formulation is usually not possible since the exponential growth of potential routes and the computational effort of computing the associated costs.

Problem Variants

Real-life vehicle routing applications generally have complex characteristics that are very diverse depending on the application area. This has led to extended problem variants that include several domain-specific extensions in terms of side constraints and objectives. As a result, several taxonomies of the large amount of problem variants have been presented including the work of Bodin (1975), Desrochers et al. (1990), Laporte and Osman (1995), Eksioglu et al. (2009) and Drexl (2012). In the context of real-world problem variants, the term *rich vehicle routing problem* has been coined (Hartl et al., 2006). These formulations consider the complex aspects of practical applications. In practice, rich side constraints exist such as driving time regulations, time-dependent travel times, or consistency considerations.

The basis for practical problem variants are often idealized formulations that are usually extensions of the archetypical VRP. As a result, several idealized problem variants have been developed and investigated separately in the literature. Typical extensions to the classical VRP include (cf. Drexl (2012)):

- time windows (Kallehauge et al., 2005),
- pickups and deliveries (Parragh et al., 2008),
- inventory routing (Moin and Salhi, 2006),
- multiple depots (Gulczynski et al., 2011),
- split-deliveries (Archetti and Speranza, 2008),
- heterogeneous fleets (Choi and Tcha, 2007), and
- periodic routing (Francis et al., 2008)

Time windows are a commonly used extension of basic vehicle routing formulations. An overview book with an emphasis to exact methods is presented by Kallehauge et al. (2005), Bräysy and Gendreau (2005a) survey

construction and local search methods while Bräysy and Gendreau (2005b) give an overview of metaheuristic approaches.

Many different variants of vehicle routing problems include time windows. For illustrative purposes the capacitated vehicle routing problem with time windows (CVRPTW) is presented here by adding time windows to the standard CVRP. The CVRPTW can be defined as an extension of the vehicle flow model presented in the previous section[2]:

$$\textit{minimize} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.8}$$

$$\textit{subject to} \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{2.9}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{2.10}$$

$$\sum_{i \in V} x_{i0} = K, \tag{2.11}$$

$$\sum_{j \in V} x_{0j} = K, \tag{2.12}$$

$$x_{ij}(\omega_i + s_i + t_{ij} - \omega_j) \leq 0 \qquad \forall i, j \in V, \tag{2.13}$$

$$a_i \leq \omega_i \leq b_i \qquad \forall i \in V, \tag{2.14}$$

$$x_{ij}(Q_i + q_j - Q_j) \leq 0 \qquad \forall i, j \in V, \tag{2.15}$$

$$Q_i \leq Q \qquad \forall i \in V, \tag{2.16}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{2.17}$$

In addition to the basic CVRP formulation, for each location $i \in V$ a time window $[a_i, b_i]$ is given which indicates that service has to start not earlier than a_i and not later than b_i. Additionally a service time s_i is given and a transition time t_{ij} between two locations. In the case of the depot, $a_0 = b_0 = s_0 = 0$. The temporal feasibility is ensured by constraints (2.26) and (2.27). Constraint (2.26) specifies the relationship between the time w_i the service starts at location i and the time w_j the service starts at location j. Constraint (2.27) ensures that the time windows are followed. The time window constraints prevent sub-tours implicitly. New constraints have been introduced to state capacity restrictions. After serving a customer the capacity is reduced by its demand q_i which is stated in equation (2.28). The remaining capacity Q_i of the vehicle leaving customer i cannot exceed the capacity C as stated in equation (2.29).

[2]adapted from Cordeau et al. (2002)

Models based on the archetypical CVRP formulation consider the distribution of goods from a depot to a set of customers. In contrast, *pickup and delivery problems* (PDP) involve pickups in addition to deliveries. There are various practical applications such as courier-services, automated guided vehicles, crane scheduling, or transportation services.

Different variants of pickup and delivery problems are surveyed by Parragh et al. (2008). It can be distinguished between two problem classes: vehicle routing problems with back-hauls and pickup and delivery vehicle routing problems. The former deals with back-hauls from customers to the depot while the latter considers transportation between locations. In their work, a detailed taxonomy is presented and fourteen variants are covered.

Two classical variants of PDP that are relevant for further investigations in this thesis are the classical pickup and delivery problem (PDP) and the dial-a-ride problem (DARP). These two variants deal with paired transportation requests between locations. In the case of the PDP, the objective is to transport goods between paired locations with minimum costs while the DARP deals with passenger transportation and especially considers service quality. The pickup and delivery problem with time windows (PDPTW) is a commonly used variant. It can be formulated as a vehicle flow model[3]:

$$\textit{minimize} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.18}$$

$$\textit{subject to} \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{2.19}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{2.20}$$

$$\sum_{i \in V} x_{i0} = K, \tag{2.21}$$

$$\sum_{j \in V} x_{0j} = K, \tag{2.22}$$

$$x_{ij}(\omega_i + s_i + t_{ij} - \omega_j) \leq 0 \qquad \forall i, j \in V, \tag{2.23}$$

$$a_i \leq \omega_i \leq b_i \qquad \forall i \in V, \tag{2.24}$$

$$x_{ij}(Q_i + q_j - Q_j) \leq 0 \qquad \forall i, j \in V, \tag{2.25}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\} \qquad \forall i \in V, \tag{2.26}$$

$$\sum_{i \in V} \sum_{j \in V} x_{ij} \leq |S| - 2 \qquad \forall S \in P, \tag{2.27}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A. \tag{2.28}$$

[3]adapted from Ropke et al. (2007)

There are many similarities to the previously presented CVRPTW model. The aim is minimization of the total costs (2.31) while constraints (2.32) and (2.33) ensure that each customer is visited exactly once and constraints (2.34) and (2.35) that each tour starts and ends at the depot. Constraints (2.36) and (2.37) specify the time window restrictions. The capacity constraints (2.38) and (2.39) are different in this model, since the demand q_i of a delivery customer is negative while the demand of a pickup customer is positive.

The use of precedence constraints ensure that each delivery customer is visited after the pickup customer by the same vehicle which have been introduced by Ruland and Rodin (1997). For that purpose, the set P is defined as a set of all node subsets $S \in V$ that represent a path from and to the depot and at least one precedence constraint is violated (e.g., a delivery location is visited in a path without visiting the corresponding pickup location first). If such a path is selected, constraint (2.40) would be violated.

As pointed out by Parragh et al. (2008), dial-a-ride problems usually either differ by considering passenger inconvenience in terms of a different objective function or in terms of additional constraints. For instance, the total throughput time can be limited by an operational constraint or can be minimized directly as a main objective.

Inventory routing problems (IRP) are quite different to the other presented formulations. Not demands, but consumption rates are given for each customer. At each customer, a certain storage capacity is given for the goods. The planning is performed over a certain multi-period horizon and each period the vendor is responsible for delivering a sufficient amount of products such that no stock-outs occur. The aim is to minimize distribution costs while maintaining a certain service level and considering storage capacities.

Bell et al. (1983) were the first to formulate an IRP in connection with the distribution of industrial gases. Since then, diverse variants and applications have been presented. Moin and Salhi (2007) give a logistical overview while Andersson et al. (2010) focus on industrial aspects and survey different approaches with respect to classification criteria depending on the considered variant. The different modeling criteria include time, demand, topology, routing, inventory and fleet aspects (cf. Andersson et al. (2010)).

One possibility to model an IRP is to divide the decisions into a planning phase and a routing phase. In the planning phase, it is decided what amount should be delivered to each customer on what day. After that decision has been made, a detailed route plan has to be derived for each day. Models frequently used for the routing and scheduling phase are standard VRP variants such as the CVRPTW (cf. Campbell and Savelsbergh (2004)).

A basic model for the planning phase of the inventory routing problem

can be formulated as following[4]:

$$minimize \sum_{t} \sum_{r} c_r x_r^t, \tag{2.29}$$

$$subject\ to\ LL_i^t \leq \sum_{1 \leq s \leq t} \sum_{i \in r} d_{ir}^s \leq UL_i^t \qquad \forall i, \forall t, \tag{2.30}$$

$$\sum_{i \in r} d_{ir}^t \leq Q x_r^t \qquad \forall r \in R, \forall t, \tag{2.31}$$

$$\sum_{r \in R} T_r x_r^t \leq M \qquad \forall t, \tag{2.32}$$

$$x_r^t \in \{0, 1\}. \tag{2.33}$$

The general aim of this basic model is to minimize the total distribution costs over all routes that are executed on each day t of the total planning horizon $t = \{1, .., h\}$ (2.42). The variable c_r indicates the costs of executing a certain route and x_r^t is a binary variable that selects route r to be executed on day t. In terms of delivered amounts, LL_i^t specifies a lower bound and UL_i^t an upper bound on the volume that has to be delivered to customer i by day t. The accumulated delivered amount must match at least than the lower bound to prevent stock-outs and not exceed the upper bound to consider storage capacities (2.43). Each of the routes cannot exceed the vehicle capacity Q which is ensured in constraint (2.44). The toal required time on a certain day cannot exceed the fleet size M as specified in equation (2.45). Each vehicle can perform multiple tour each day t while the length of the tour is given as a fraction of the maximum daily operation time ($T_r \in [0, 1]$).

Exact and Heuristic Algorithms

Both exact and heuristic methods have been successfully applied in practice and are constantly improved on well-known benchmark instances of different VRP variants. An overview of exact and heuristic algorithms is given in the following. This overview is mainly based the surveys of Laporte (2009), Doerner and Schmid (2010), and Baldacci et al. (2012).

As pointed out by Laporte (2009), exact methods have evolved to highly sophisticated methods based on mathematical programming in the last 40 years. Early approaches such as Christofides and Eilon (1969) mainly developed basic branch-and-bound algorithms. Also dynamic programming has been applied based on the formulation of Eilon et al. (1971), however this research direction has not received much attention since.

[4]Campbell and Savelsbergh (2004)

The currently most successful techniques are based on a set-partitioning formulation and advanced column generation algorithms (cf. Baldacci et al. (2012)). Fukasawa et al. (2006) apply a branch-and-cup-and-price algorithm to a set-partitioning formulation that has been augmented with various additional valid inequalities. The algorithm proposed by Baldacci et al. (2008) is also based on an augmented set-partitioning formulation and incorporates lower bounds calculated by three different heuristics into a branch-and-cut approach.

Due to recent advances, benchmark instances with up to about 100 customers have been solved to optimality using exact algorithms. For example, the last two instances of the well-known 56 VRPTW benchmark instances defined by Solomon (1987) that consist of 100 customers have been solved recently (cf. Baldacci et al. (2011) and Ropke (2012)).

Even though much progress has been made in the field of exact methods, Laporte (2009) points out that VRP variants remain difficult to solve in practice, especially when considering large problem sizes and when complex side-constraints are involved. As a result, a multitude of heuristic algorithms have been developed which provide a compromise between solution quality and runtime. Laporte (2009) distinguishes between classical heuristics and metaheuristics.

Classical heuristics are mostly based on solution construction that is often combined with a simple improvement phase. The most important classical heuristics are the savings, cluster-first route-second and intraroute/interroute improvement heuristics (cf. Laporte (2009)).

One prominent example that is still frequently referenced today is the savings heuristic developed by Clarke and Wright (1964). It starts with a set of tours where a single customer is assigned to each vehicle. Then it iteratively merges the tours starting with the ones that result in the largest saving. Improvements to this basic scheme include the work of Altinkemer and Gavish (1991) and Wark and Holt (1994).

Cluster-first, route-second heuristics first assign the locations to vehicles and then, for each vehicle, build an associated route. One approach is assembling a set of vehicle routes to a feasible solution by means of a set-partitioning model. For instance, the sweep algorithm proposed by Gillett and Miller (1974) generates non-overlapping routes by iteratively adding locations in a circular manner as long as the constraints are satisfied. A construction heuristics hybridizing the sweep heuristic with other insertion procedures is the push forward insertion heuristic proposed by Solomon (1987). Another approach is the generation of initial seed points and then building the clusters by solving a generalized assignment problem as proposed by Fisher and Jaikumar (1981). An interesting aspect of cluster-first route-

second heuristics is the decomposition into sub-problems as also generally applied by human dispatchers.

Intraroute and interroute improvement heuristics rely on local search and neighborhood operations. They are often used as a second stage after a construction heuristic has been applied to generate an initial solution. Commonly used intraroute operations are adapted from the TSP literature such as the 2-opt or the OR-opt operation. Interroute operations usually involve the relocation of customers between routes. An example is the transfer scheme of Thompson and Psaraftis (1993).

On the contrary to specialized problem-specific heuristics, *metaheuristics* are general search-strategies that guide problem-dependent operations. They are based on an efficient interplay between diversification and intensification. There is an ever-growing body of literature applying metaheuristics to diverse VRP variants. Reviewing successful search strategies, Cordeau et al. (2002) conclude that the metaheuristics available are highly accurate. However, there is a tendency to over-engineered and over-specialized methods pointing out the need for flexible and simple heuristics.

There are some general success factors that can be identified within the wide range of metaheuristics that are useful for solving large-scale vehicle routing problems. In general, a distinction can be made between local search based methods and population based methods.

Local search based methods iteratively explore the solution space by moving from the current solution to a neighboring solution at each step. An essential challenge is not getting stuck in a locally optimal solution and there are various metaheuristic search strategies to avoid that issue. In the context of vehicle routing, especially tabu search (Glover, 1989), variable neighborhood search (Mladenović and Hansen, 1997), and large neighborhood search (Pisinger and Ropke, 2010) have received extensive attention (cf. Laporte (2009)).

A prominent tabu search variant that has been applied to diverse variants of the VRP is the unified tabu search initially proposed by Cordeau et al. (2001). It is based on a relocate neighborhood where a single customer is relocated to another route. An important success factor of this algorithm is the adaptive penalty relaxation to allow infeasible solutions. Whenever a constraint is violated, the penalty is increased otherwise it is decreased. Additionally, a long-term memory is applied for diversification to penalize similar solutions. Other successful tabu search variants include the work of Taillard (1993), Gendreau et al. (1994), and Toth and Vigo (2003). A general success factor of tabu search implementation is the use of the search history in terms of short-term, mid-term and long-term memory.

Variable neighborhood search contains a shaking and improvement phase.

In the shaking phase, a switch is performed to increasingly large neighborhoods. Within this neighborhood a local optimum is obtained in the improvement phase. This concept allows visiting different locally optimal solutions. Applications to VRP variants include the work of Polacek et al. (2004) and Kytöjoki et al. (2007). A successful concept is changing the assignment of customers to vehicles during the shaking phase and improving the routes during the improvement phase leading to an interplay between clustering and routing.

Large neighborhood search is based on destroy and repair operations. These operations range from very small changes such as the removal of a customer to very large one such as the removal of whole tours. Also heuristic destroy procedures are used such as cluster or time-oriented removal. Reconstruction can occur using insertion heuristics or local improvement procedures. An adaptive variant was proposed and applied to a VRP by Ropke and Pisinger (2006) where the neighborhoods are chosen with a probability proportional to their success. In general, a key success factor of large neighborhood search is the exploration of potentially large neighborhoods allowing an efficient interplay of diversification and intensification.

Population based methods work with a set of solutions at the same time as opposed to local search methods who consider the improvement of a single incumbent solution. The most prominent method of this paradigm are genetic algorithms (Holland, 1975) that apply the principles of natural evolution to optimization. However, as pointed out by Vidal et al. (2012), most successful search strategies for the VRP involve local search. As a result, genetic algorithms have been mainly applied in combination with a local improvement procedure leading to memetic algorithms. Recently, Vidal et al. (2012) proposed a memetic algorithm with diversity control that outperforms existing approaches on several VRP variants. In general, diversity management in combination with local search seems to be an important success factor for genetic search.

The combination of heuristic and exact methods has led to *Matheuristics* that combine mathematical programming techniques with heuristic search strategies. A survey on matheuristics for VRP variants is given by Doerner and Schmid (2010). They can be basically categorized into set-covering, local branching and decomposition based approaches.

Integrative set-covering approaches (e.g, Archetti and Speranza (2008)) combine heuristic route generation with the exact solution of the covering problem while in collaborative set-covering approaches (e.g, Schmid et al. (2009)) information between the heuristic and the exact method is exchanged. Local branching procedures (e.g, Schmid et al. (2010)) use metaheuristics for branching decisions at a strategic level in an exact solver by fixing certain de-

cision variables. Decomposition approaches are based on decomposing a large problem into smaller sub-problems that can be solved exactly. Examples are a hierarchical decomposition through heuristic ruin and exact recreate operations (De Franceschi et al., 2006) or the solution of certain sub-problems such as the exact determination of timing a route (Hashimoto et al., 2008).

Summarizing, there is a large body of research on static vehicle routing problems. Advanced methods are available for practically relevant variants. The methodology is still improved and larger and more complex problems can be solved using hybrid algorithms combining several successful solution methods.

2.1.2 From Static to Dynamic Vehicle Routing

Practical applications where the evolution and quality of information is an important aspect in combination with advances in the fields of operations research and telematics have led to an increasing interest on dynamic vehicle routing problem variants. Dynamic vehicle routing deals with environments, where the information available to the decision maker evolves during the planning process. Psaraftis (1988) gives a definition on the dynamic vehicle routing problem:

> [...] in a "dynamic" vehicle routing problem, inputs may (and, generally, will) change (or be updated) during the execution of the algorithms and the eventual execution of the route. Actually, algorithm execution and route execution are processes that evolve concurrently in a dynamic situation, in contrast to a static situation in which the former process clearly precedes (and has no overlap with) the latter.

Even though the field can build on the extensive research on static VRP variants, there are fundamental differences concerning modeling and optimization approaches. This mainly stems from the fact, that not all information is available at the beginning of the planning process and the resulting uncertainty about future events. As a result, newly arriving information might require plan changes or render previously made decisions sub-optimal.

Psaraftis (1988) outlined the main differences between static and dynamic vehicle routing as following:

1. Time dimension is essential: Since new information arrives during the planning process, scheduling is always an important aspect of dynamic vehicle routing. In contrast, static variants such as the CVRP that do not contain time restrictions do not involve scheduling.

2. The problem may be open-ended: In a dynamic situation the planning horizon can be unbounded which leads to the planning of open paths for the vehicles rather than closed tours.

3. Imprecise or unknown information: Information about future events may be imprecise or unknown. Probabilistic information might be available, however in some cases knowledge about future events does not exist.

4. Importance of near-term events: Due to the uncertainties associated with future events, long-term commitments are likely to become sub-optimal due to newly arriving information in the meantime. This is not the case in static vehicle routing problems where near-term and long-term events have the same weight.

5. Information update mechanisms are required: Problem inputs need to be changed as new information appears and consequent decisions have to be made efficiently.

6. Re-sequencing and re-assignment decisions: Previously made assignment or sequencing decisions can be rendered sub-optimal as new information arrives. As a result, as the input changes, the assignment of requests to vehicles or the sequencing of requests might have to be changed in order to consider the new situation.

7. The need for faster computation times: In static routing environments the aim is to obtain a best as possible, in the best case optimal, solution within reasonable computational time which in many cases can be even a couple of hours. In contrast, in dynamic environments it might not be worth solving a current situation close to optimality since the arrival of new information might render the current solution sub-optimal. Additionally, in dynamic environments faster response times are needed since decisions have to be made while the routes are executed and new information is arriving.

8. Indefinite deferment mechanisms: Due to the focus on the optimization of near-term actions and the possible open-ended planning horizon, serving unfavorable requests might be deferred indefinitely. This phenomenon is called "starving". This might be caused by simplistic heuristics such as always serving the nearest customer. A customer in a distant geographic location might never be served. Measures must be added to mitigate that issue in that case.

9. Different objective functions: For static vehicle routing problems, the objective is usually the minimization of total costs. This might not be feasible in the dynamic case since the planning horizon might be unbounded or not all information is known at a given time or stochastic information is available. In some cases, the objective function has to be extended to avoid undesirable behavior. For instance, a penalty could be added for indefinite deferment of requests.

10. Different time constraints: Due to the possibility that newly arriving information can lead to a situation where a hard deadline of a customer cannot be met, violating time constraints may be regarded a better alternative then denying the service to the customer. In that sense, time constraints are often considered softer then in static vehicle routing variants.

11. Lower flexibility to vary fleet size: When solving a static routing problem, there is usually some time span before the routes are executed. This allows more flexibility to add an additional vehicle in order to meet the customer demands. In a dynamic setting there is usually less flexibility to add more capacity in real-time.

12. Importance of queuing considerations: If the customer demand exceeds the capacities over a longer time span, the system will become congested. At such a state, it becomes impossible to achieve a good service quality by using typical routing strategies. Psaraftis (1988) suggested the application of queuing theory, however according to Larsen et al. (2007) the applications are still scarce.

2.1.3 Dynamically Arriving Information

As stated in the previous section, the main challenge in dynamic vehicle routing environments is the uncertainty associated with dynamically evolving information. Several input values can be subject to changes over time in a dynamic vehicle routing setting.

As summarized by Richter (2005) and Larsen et al. (2007), the following input values are typical examples of dynamically evolving information:

- Arrival of new requests: The appearance of new requests during the planning period is an extensively researched variant of dynamic VRPs. At the beginning of the planning process not all customer locations are known yet. The algorithmic strategies have to account for the fact that the routes must be flexible to incorporate newly arriving requests. In

the best case they can be scheduled efficiently in the existing routes, in the worst case a complete re-planning might be necessary.

- Service cancellations: Customers might cancel their request. One example would be the cancellation when deadlines are missed or the customer is waiting too long to be serviced.

- Update of demands: In some cases the customer location might be already known, but an uncertainty might be associated according to the exact amount of demand. For instance, the exact demand might be revealed when the vehicle arrives at the location in the context of heating oil distribution.

- Change of time windows: In some application areas, the time windows the vehicle is allowed to arrive can be changed. An example are pickup or delivery operations that are dependent on good availability.

- Change of capacities: Due to unforeseen events such as vehicle breakdowns the available capacity might decrease during the planning process.

- Dynamic travel times: Sudden traffic disturbances such as traffic jams caused by accidents can occur and might require a change of the vehicle routes.

Psaraftis (1995) provides a taxonomy for the characterization of the inputs of dynamic vehicle routing problems. Four important concepts are highlighted: the evolution of information, the quality of information, the availability of information and the processing of information.

In terms of the evolution of information Psaraftis (1995) distinguishes between static and dynamic inputs. Static inputs are known already before the planning process and are not updated over time while dynamic information is revealed or updated. It is important not to confuse dynamically evolving information with time-dependent inputs. One example are time-dependent travel times which are fixed from the beginning and are thus considered a static input. On the contrary, real-time traffic information would be considered as a dynamic input.

Regarding the quality of information Psaraftis (1995) categorizes the inputs of a dynamic VRP as known-deterministic / forecast / probabilistic and unknown. These attributes are defined for a given time in the planning process. For instance, once a customer request appears, its information quality changes from unknown to known-deterministic. As pointed out by Larsen

et al. (2007), in many applications the information quality is higher for near-term than for distant events. Deterministic inputs are known with certainty and do not change over time. Other inputs may only exist as forecasts and become updated over time. If they follow certain probability distributions, they are considered as probabilistic. Other inputs may be completely unknown at given times.

The availability and the processing of information influence are important factors in the design of decision support systems. The fact that certain information is only available locally requires a decentralized processing. If the relevant information is available globally to a central decision making unit, a centralized system can be created. In general, due to advances in the field of telematics, more and more information is available globally as noted by Larsen et al. (2007). It could be also by design that certain decisions such as selecting alternative routes in the case of a traffic jam are delegated from the central dispatcher to the drivers.

Based on these considerations two important topics are highlighted. The degree of dynamism measure characterizes dynamic vehicle routing environments in terms of the evolution of information. The use of probabilistic and forecast data is incorporated in dynamic and stochastic models.

Degree of Dynamism

For the performance evaluation of dynamic vehicle routing systems, the characterization in terms of dynamically arriving information is an important aspect. For this purpose, Larsen (2000) motivated the degree of dynamism as a single performance measure that is based on the ratio of dynamically arriving in relation to static information and also on the time this information is revealed.

It should be noted that the measure focuses solely on dynamically arriving requests. As pointed out by Zaepfel and Vogl (2010) this is a common approach to characterize the dynamism of the system but there are other influence factors that should be considered which are summarized by Richter (2005). The author is not aware of approaches that have adapted the degree of dynamism measures to consider additional aspects.

Originally the degree of dynamism (dod) measure for dynamic systems was introduced by Lund et al. (1996). They define the ratio between total (n_{tot}) and dynamic (n_{imm}) requests as:

$$dod = \frac{n_{imm}}{n_{tot}} \tag{2.34}$$

For instance, if a-posteriori $n_{tot} = 100$ requests were placed during the

planning process and $n_{imm} = 10$ arrived dynamically while 90 were known in advance the degree of dynamism is $dod = 10\%$.

The *dod* measure has, however some limitations because it does not consider when the dynamic requests actually arrive. It has been extended by Larsen (2000) to include the information about the arrival time of the immediate requests. This measure is defined as the effective degree of dynamism (*edod*) which considers the arrival time of each dynamic request (t_i) in relation to the planning horizon (T):

$$edod = \frac{\sum_{i=1}^{n_{imm}} \frac{t_i}{T}}{n_{tot}} \tag{2.35}$$

This implies, that the later the dynamic requests arrive the higher the *edod* measure will get. In a pure static problem environment, all requests are known at time $t_i = 0$ which means the measure will be $edod = 0$. Highly dynamic environments are characterized by many late-arriving requests. The *edod* measure thus reflects the temporal distribution (cf Larsen et al. (2007)).

Extending this concept, Larsen (2000) defined the $edod_{tw}$ measure that also includes time windows. It takes into account the reaction time available to the planner in terms of due dates specified by the customers. The definition is based on the reaction time r_i that is available between the appearance of the request at time t_i and the closing of the time window at time b_i:

$$edod_{tw} = \frac{1}{n_{tot}} \sum_{i=1}^{n_{imm}} (\frac{T - (b_i - t_i)}{T}) = \frac{1}{n_{tot}} \sum_{i=1}^{n_{imm}} (1 - \frac{r_i}{T}) \tag{2.36}$$

The average available reaction time is inversely proportional to the value of the $edod_{tw}$ measure. As Larsen et al. (2007) note, a long reaction time is generally desirable because more computation time is available for processing this new request.

Stochastic Information

Dynamic vehicle routing problems are characterized by incomplete information that is revealed gradually during the planning process. As identified by Psaraftis (1995), this information can be known with some uncertainty however. In many practical applications stochastic information about future requests is available or can be obtained by means of historical data or by creating probabilistic models (cf. Bent and Van Hentenryck (2004b)). As pointed out by Zaepfel and Vogl (2010), in the past approaches for dynamic VRPs did not incorporate stochastic information, however recently there is an increased interest in dynamic and stochastic models.

Pillac et al. (2012a) pointed out the significance of the quality of information for the classification of vehicle routing problems by distinguishing between deterministic and stochastic input. Combined with the evolution of information, distinguishing between static and dynamic input, a taxonomy of four different classes of vehicle routing problem results.

Static and deterministic routing problems correspond to classical static variants. All information is known beforehand with certainty and does not change over time. As noted by Pillac et al. (2012a) and many other authors, in many practical applications this simplifications are not valid assumptions.

Static and stochastic problem formulations incorporate random variables which realizations are revealed before the execution of the routes. Reviews on the field of stochastic vehicle routing have been provided for instance by Gendreau et al. (1996) and Cordeau et al. (2007). The three most commonly studied cases are stochastic customers, stochastic demands and stochastic times (cf. Cordeau et al. (2007)). The planning occurs by means of a two-stage process. In a first stage, a plan is created that accounts for the stochastic variables. Then the realizations of the variables are revealed and small changes to the plan are made in terms of recourse actions such as skipping the customers that did not place orders.

In dynamic and deterministic vehicle routing problems, inputs are considered to be completely unknown and to appear over time while the routes are executed. As Zaepfel and Vogl (2010) note, the fields of stochastic vehicle routing and dynamic vehicle routing have evolved separately in past. A combination of the two fields leads to dynamic and stochastic vehicle routing problems that consider stochastic information which is available about unknown inputs. A main difference to static stochastic problems is the fact that the plan is updated as the routes are executed while in static models they are created a-priori.For some scenarios it might make sense to create an a-priori plan (e.g., in recurrent situations) while for other applications a real-time optimization might be beneficial.

2.1.4 Objectives and Performance Evaluation

For classical variants of static vehicle routing problems, the most commonly used objective is the minimization of total distribution costs which consist of fixed costs for the required fleet as well as variable costs for the driven distance. There are different objectives for evaluating the performance of dynamic vehicle routing systems depending on the application area.

Larsen et al. (2007) pointed out the three most important elements: minimization of distribution costs, maximization of the service level, and throughput optimization. These are conflicting objectives since raising the service

level usually implies an increase in costs. Also throughput optimization and reducing the wait time can conflict since the focus on a fast response time for some customers may decrease the ability to serve as many customers as possible and lead to starvation of other customers.

An additional aspect that comes into play for dynamic VRPs is missing information. When evaluating a given situation, possible future information has to be considered. A solution that is optimal at a given situation might be rendered sub-optimal by additional arriving information.

Considering all different aspects of dynamic vehicle routing systems, the objective of a dynamic VRP is usually a combination of several factors including distribution costs, service level, throughput and flexibility with regard to newly arriving information (cf Zaepfel and Vogl (2010)).

For the evaluation of the performance in a dynamic vehicle routing environment, there are basically two different approaches as outlined by Ghiani et al. (2003). Dispatching and routing algorithms can be cosidered analytically if certain simplifying hypothesis can be assumed such as Poisson distribution of the dynamic variables or uniform distribution of the appearing customers. In cases where these assumptions are not suitable, an empirical evaluation has to be performed by means of discrete-event simulation.

For the analytical investigation of routing and dispatching policies, the competitive analysis framework is commonly applied which was initially defined by Sleator and Tarjan (1985) for production planning. In a competitive analysis, the worst-case performance of an online algorithm is compared with an optimal algorithm that has access to all relevant information beforehand. In that context, the concept of the competitive ratio is important:

$$C_A(i) \leq c * C_{Opt}(i), \forall i \in I \tag{2.37}$$

An online algorithm A is denoted c-competitive if its performance is maximal c times worse than an optimal offline algorithm C_{Opt} on all possible problem instances $i \in I$.

A survey is provided by Jaillet and Wagner (2008). Early work includes Bertsimas and Van Ryzin (1993) where they analyzed worst-case bounds for single- and mutli-vehicle cases of the traveling repairman problem and derived optimal policies for light and heavy traffic situations. More recent work includes Jaillet and Wagner (2006) who studied competitive ratios for online traveling salesman as well as traveling repairman problem and Angelelli et al. (2007) who considered multi-period problems.

As Larsen et al. (2007) note, the inclusion of practically relevant constraints such as time-windows is usually to complex for a consideration using competitive analysis. An alternative are empirical studies using discrete-

event simulations. When considering practically relevant variants with complex side constraints this is a commonly used approach. The concept of the competitive ratio has been adapted by Mitrović-Minić et al. (2004) to an empirical analysis on both a dynamic and static variant of the problem by comparing the average observed algorithm performance. That way, conclusions can be drawn on the value of information.

2.1.5 Categorization and Application Areas

Based on the degree of dynamism definition and the primary optimization objective, Larsen et al. (2002) presented a classification framework for dynamic vehicle routing problems. In that scheme, different application areas can be categorized. This is especially important, since the developed methods differ dramatically depending on the characteristics of the problem environment.

The first dimension of categorization is the degree of dynamism. Weakly dynamic systems are characterized by a low number of dynamically appearing information that is usually not more than 20%. Moderately dynamic systems already contain a significant amount of immediately arriving information. Strongly dynamic system typically have a degree of dynamism of 80% and greater.

The second dimension of this categorization is the main system objective of the application area. Larsen et al. (2007) note, that the applications are located around the diagonal of these two dimensions. In weakly dynamic systems the main objective is close to the one of a static routing problem which is usually the minimization of transport costs. In strongly dynamic systems the main goal is typically the minimization of response time. However, as pointed out by Zaepfel and Vogl (2010) in terms of revenue maximization, focusing on a single target is not a way to success. As an example, when minimizing the response time, also costs have to be considered while the main target remains the service quality.

An overview of different application areas and their classification according to the presented scheme is given in Figure 2.1 based on the work of Larsen et al. (2007). Additional applications such as city logistics, home deliveries, or air taxis can be found in a recent survey given by Pillac et al. (2012a).

The distribution of goods such as groceries, heating oil or liquid gas to a large number of households is usually characterized by a low frequency of changes and a relatively large reaction time. There is usually a fixed set of routes that are carried out daily derived from customer subscriptions. However, a small percentage of customers may require additional deliveries or cancel their order due to exceptional situations. Another example of a weakly dynamic system is the transportation of elderly and handicapped

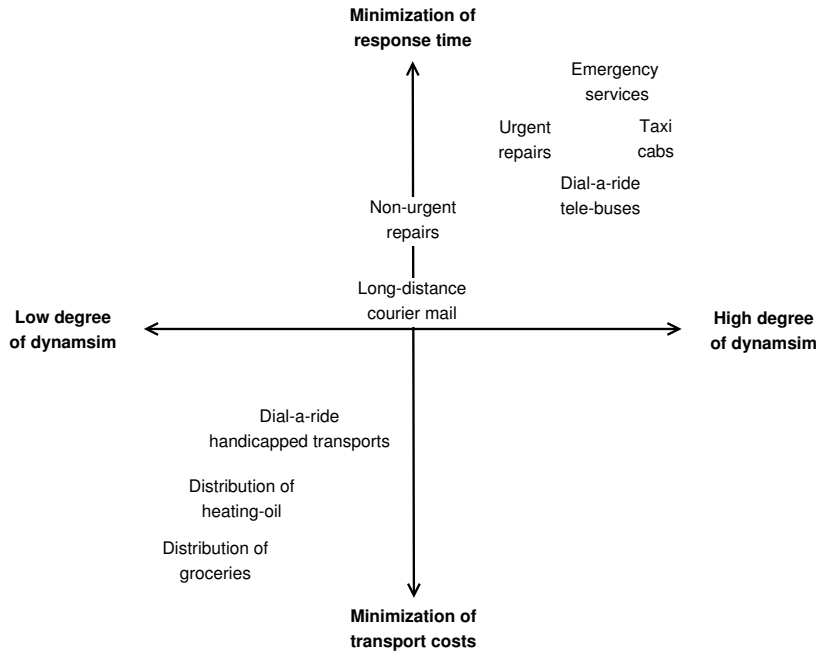

Figure 2.1: Classsification of dynamic vehicle routing problems according the degree of dynamism and primary objective (cf. Larsen et al. (2007))

persons who generally book their trips a few days ahead. A possible approach in the context of weakly dynamic systems is to model the problem either as a stochastic vehicle routing problem with recourse actions or by means of re-optimization leading to a dynamic model. The main optimization target is the minimization of transportation costs since only few immediate information arrives during the plan execution.

In moderately dynamic systems there is already a significant amount of dynamically arriving information, however the routing algorithm should still take the static information into account sufficiently. As a result, there is often a conflict between transport cost and response time minimization in that context. Examples for moderately dynamic systems are non-urgent traveling-repairman problems such as repairing appliances or long-distance couriers. Solution procedures must be characterized by a fast response time since an adaption of the plan will occur quite often as new information arrives.

For strongly dynamic systems the focus is laid on a fast response time. A typical example are emergency services where no information is known in advance and the main objective is a minimization of waiting time. Another example are taxi cab services or dial-a-ride tele-buses where little or no requests are known in advance. Urgent repairs such as car breakdown services

also fall in this category. Previous work has shown, that considering future requests can improve the solution quality significantly in this context. Examples are the relocation of vehicles in central areas or waiting where new requests are likely to occur.

As Larsen et al. (2007) conclude, the system objective and the degree of dynamism should be considered when creating algorithmic approaches for dynamic vehicle routing problems. Obviously, optimizing emergency service operations is quite different than optimizing the distribution of groceries. As a result, several specialized solution methods exist which will be discussed in the following considering the characteristics of dynamic problem environments.

2.2 Solution Methods

Dynamic vehicle routing problems require specialized solution methods due to the challenges that are inherent to dynamically changing environments. As noted by Pillac et al. (2012a), these aspects increase the complexity of decision making compared to static routing problems. Additional degrees of freedom have to be considered as well as finding a compromise between the reaction time and the decision quality. As a consequence, the methods applied in the field of dynamic vehicle routing have to deal with the inerent dynamism and resulting uncertainty of the problem environment.

Several developments have accelerated the evolution of solution methods for dynamic vehicle routing problems. Ghiani et al. (2003) identified the increase in computational power, accurate metaheuristics and the advances in the field of parallel computation as main driving factors of methodological developments. At the same time, advances in information and communication technologies enabled real-time fleet management in more and more application areas which lead to an increasing interest in solving the underlying dynamic routing problems. Pillac et al. (2012a) state, that especially in the last decade a growing body of research can be observed.

Surveys led to an organization and categorization of the approaches presented in the literature. Recently Pillac et al. (2012a) categorized solution methods according to the evolution and quality of information. Zaepfel and Vogl (2010) focused on the adaption to changes and distinguishing between event-based adaption and a-priori planning with dynamic adaption. A broad view on advances and research challenges was provided by Larsen et al. (2008). Ichoua et al. (2007) focused on the design of planned routes in problem environments with dynamically arriving customers. Jaillet and Wagner (2008) outlined several analytical studies based on queuing theory. Parallel

computation strategies were investigated by Ghiani et al. (2003).

Based on these surveys, several solution methods will be outlined focusing on four important aspects:

- By combining queing theory and dynamic routing, methods based on dynamic policies reactively dispatch vehicles as new information arrives
- Building on the extensive research on static VRPs, several solution methods have been derived by the adaption of static algorithms
- Methods for considering future events account for dynamically arriving information
- Parallelization strategies play an important role because of the limited reaction time in dynamic environments

2.2.1 Dynamic Policies

Dynamic routing policies do not focus on planned routes but rather view the vehicle as a mobile server and decide what request to serve next among a queue of pending requests (cf. Ichoua et al. (2007)). Using queuing theory, several policies have been investigated analytically under simplifying assumptions such as unlimited capacity or uniformly distributed requests. On the other hand, policies have been derived and tested empirically for complex real-world situations.

Several policies for the dynamic traveling repairman problem (D-TRP) have been investigated analytically using queuing theory by Bertsimas and Van Ryzin (1991). Underlying assumptions are independently Poisson distributed arrival rates, service times of customers that are uniformly distributed in an Euclidean plane, and a server with unlimited capacity. The aim is to derive a strategy that minimizes the total system time.

The following basic policies have been investigated:

- First come first serve (FCFS): Requests are serviced in the order they arrive which corresponds to a first in first out principle (FIFO). If there are no unserviced demands, the vehicle waits at the current location.
- Stochastic queue media (SQM): The customers are serviced in a FCFS manner, however the vehicle returns to the geographic median after completing each service.
- Partitioning (PART): The geographic area is divided into m sub-regions, where m is a parameter of the strategy. Within a region, the FCFS

principle is applied while there are still requests left. Then the vehicle moves to an adjacent region where requests are queued. This principle is repeated until all requests have been served.

- Space filling curve (SFC): Requests are served as they appear in repeated clockwise sweeps while the depot is visited once per each sweep.
- Nearest neighbor (NN): Whenever the vehicle finished serving a request, it continues to the geographically nearest queued request.
- Traveling salesman (TSP): A batching of requests occurs and after a certain number has been reached, the resulting TSP is solved to optimality. Strictly speaking, this is not a pure dispatching policy but involves planning of routes.

An analytical study was performed by Bertsimas and Van Ryzin (1991) both for light and heavy traffic situations by deriving worst-case performance bounds in terms of competitive ratios as well as simulation. In the case of light traffic, the SQM policy has been shown to be optimal. However, the FCFS policies become unstable for increasing traffic intensities. Papastavrou (1996) has later defined the generation (GEN) policy, that basically combines the SQM and TSP strategies and behaves well in low-traffic as well as in heavy-traffic situations.

The work of Bertsimas and Van Ryzin (1991) has been extended to a multi-vehicle case by Bertsimas and Van Ryzin (1993) and to a single-vehicle pickp and delivery problem by Swihart and Papastavrou (1999). Larsen et al. (2002) have empirically compared the strategies for D-TRP instances with varying degrees of dynamism that consist of both static and dynamic requests. The main finding was, that the routing costs increase linearly with the degree of dynamism and the nearest-neighbor policy performed best for distance minimization. Ausiello et al. (2001) have considered the traveling salesman problem on the metric space and derived an optimal policy.

Recently, some approaches have derived dispatching policies empirically by means of simulation optimization. Beham et al. (2009b) have evolved complex dispatching rules for the dynamic dial-a-ride problem (D-DARP) using a linear combination of different dispatching information and by tuning the parameters by means of simulation optimization. van Lon et al. (2012) and Vonolfen et al. (2013c) have evolved complex policies for D-DARP instances by means of genetic programming. These policies are adapted to certain scenarios and combine several simple policies such as NN or FCFS to decision trees.

2.2.2 Adaption of Static Algorithms

Building on the extensive research on solving static vehicle routing problems, a commonly used approach to solve dynamic variants is to adapt static algorithms. In that sense, the dynamic vehicle routing problem is regarded as a sequence of static problems and the algorithm is applied to the current situation. This procedure is carried out each time new information is revealed or within fixed time slices (frequently also called decision epochs). Successful concepts are to transfer information between the optimization of the individual situations (cf. Pillac et al. (2012a)) and the focus on short-term events in a rolling horizon manner while postponing long-term events (cf. Ichoua et al. (2007)).

In terms of the adaption of static algorithms, Ichoua et al. (2007) distinguish between local update and re-optimization procedures. Local update methods build a set of a-priori routes from the information that is known at the beginning of the planning process and newly arriving information is incorporated using fast local update procedures. In contrast, re-optimization methods solve each situation from scratch. The ability to completely reconsider decisions where no commitment was made yet such rescheduling the pending requests allows utilizing a larger optimization potential but generally leads to a longer reaction time. Hybrid approaches combine the strength by using local update procedures for making fast decisions and applying re-optimization when computation time is available.

Local update procedures usually apply simple insertion heuristics to incorporate dynamically arriving requests into the current planned routes. For instance, Madsen et al. (1995) have proposed an insertion heuristic for the dispatching of repairmen in a highly dynamic environment. Insertion heuristics are also used in hybrid approaches to quickly determine if a request can be incorporated in the current set of routes and, after making the decision whether to accept the request, running an re-optimization procedure.

Both exact and heuristic methods have been applied within re-optimization approaches, however most re-optimization approaches are based on heuristics as Pillac et al. (2012a) and also other authors noted. This stems from the fact that the response time is a critical factor in dynamic routing environments. In many cases it might not be worth the computational effort to solve a given situation to optimality since newly arriving information can render the previously made decisions sub-optimal. As a result, a balance must be achieved between the computational effort and the solution quality.

A broad range of literature has focused on the application of metaheuristics for re-optimization. Among others, genetic algorithms (GA), ant colony optimization (ACO), and tabu search (TS) have been applied. Common

among successful approaches is an incorporated methodology for maintaining information about good solutions. This observation has been generalized to the multiple plan approach (MPA). Most presented approaches are not pure re-optimization approaches but hybridized with local update procedures.

When applying GAs to dynamic routing problems, the population is usually kept for the whole optimization process and updated as new information arrives (cf Pillac et al. (2012a)). Information update mechanisms are provided to transfer information throughout the plan revisions such as inserting new customers into the solutions of the population or deleting served customers. For instance, a GA combining local update procedures with re-optimization has been applied to dynamic PDPTW instances by Pankratz (2005). In a similar manner, Haghani and Jung (2005) and Cheung et al. (2008) applied GAs to dynamic PDP. van Hemert and La Poutré (2004) extended this concept with relocation strategies for fruitful regions.

As pointed out by Ichoua et al. (2007), ACO algorithms applied to dynamic vehicle routing achieve the transfer of information between the re-optimization steps by preservation of the pheromone matrix. Similar to GAs, an update mechanism is provided to adapt the pheromone matrix to newly arriving information. This pheromone conservation strategy has been applied by Gambardella et al. (2003) and Montemanni et al. (2005) to a dynamic VRP without time windows. A matheuristic for the VRP with dynamic travel time combining ACO and dynamic programming was proposed by Chitty and Hernandez (2004).

Being a successful strategy for static formulations, several TS variants have been presented for dynamic vehicle routing problems. Gendreau et al. (1999) have applied the parallel TS with adaptive memory proposed by Taillard et al. (1997) to a dynamic VRPTW problem. The adaptive memory is used to store good partial solutions (routes) that are found during the search. New starting solutions are generated from the memory and from them, several tabu search threads are started in parallel. An information update mechanism is provided to update the adaptive memory as new information arrives. A two-stage approach combining insertion and improvement phases was followed by Attanasio et al. (2004), who adapted a parallel variant of the method proposed by Cordeau and Laporte (2003) to the dynamic DARP. The incumbent solution of each search thread is updated and the search is then continued. A similar approach is applied by Beaudry et al. (2010) to optimize patient transportation.

Generalizing from the underlying re-optimization algorithm and identifying the importance of information transfer between the optimization steps, Bent and Van Hentenryck (2004b) have presented the multiple plan approach (MPA). Multiple plans are held that are consistent with the current situation.

At each time step, the plans in the pool are updated to be compatible with the executed plan. Incompatible plans are deleted. An important aspect is the selection of the distinguished plan that is executed by the vehicles.

2.2.3 Considering Future Events

A main challenge in dynamic vehicle routing is the evolution and quality of information that influences the planning process. Decisions that are optimal at a given time might be rendered sub-optimal by arriving events such as new requests, vehicle breakdowns or service cancellations. Ichoua et al. (2007) note that human dispatchers usually consider advance knowledge such as demand patterns in the planning process. From the algorithmic perspective, solution approaches have been developed that take into account possible future events and increase the flexibility in terms of newly arriving information.

It can be distinguished between methods that are based on stochastic information about future events and methods that are based on general assumptions. Stochastic information is often derived from historical data and is incorporated in dynamic and stochastic routing problems. Pillac et al. (2012a) point out several approaches how this probabilistic knowledge about future events can be utilized in solution methods. It can be either incorporated analytically, by scenario sampling or in other specially designed strategies.

Methods based on stochastic modeling incorporate probabilistic knowledge analytically (cf. Pillac et al. (2012a)). One stream of research is to model dynamic and stochastic vehicle routing problems as Markov Decision Processes (MDP) and apply dynamic programming to maximize the expected profit. Powell et al. (1988) have solved a PDP with demand uncertainty while Kim et al. (2005) consider a VRP with dynamic travel times. To increase the scalability for larger problem sizes, Approximate Dynamic Programming uses approximation schemes to avoid full value function evaluations (cf Powell (2007)). For instance, Novoa and Storer (2009) have presented a re-optimization approach using ADP for a VRP with stochastic demands. Another stream of literature focuses on the application of methods from stochastic linear programming such as the work of Yang et al. (2004).

Sampling approaches generate scenarios which are realizations of the probability distributions. These scenarios are then solved independently and conclusions are drawn from the scenario pool. The multiple scenario approach (MSA) is an extension of the MPA that has been presented before. It has been proposed by Hentenryck and Bent (2009) as a general framework for solving online stochastic combinatorial optimization problems with

dynamic vehicle routing as one application area. Several scenarios are sampled from probability distributions of possible future demands that include both actual and forecasted requests. Bent and Van Hentenryck (2004b) applied the MSA to a VRP with stochastic customers. The executed plan has been chosen using a consensus function which selects the plan that is most similar to the others and thus represents some kind of least-commitment. Subsequent work has investigated different approaches for choosing the distinguished plan (Bent and Van Hentenryck, 2004a) and the incorporation of waiting and relocation (Bent and Van Hentenryck, 2007). A similar approach to the MSA has been presented by Hvattum et al. (2006). Sampling has also been incorporated in local search techniques such as TS (Attanasio et al., 2007) and adaptive large neighborhood search (Azi et al., 2012).

Besides these two general frameworks, several strategies have been developed that focus on certain aspects such as waiting, relocation, or increasing the flexibility to incorporate future requests.

Waiting strategies are suitable for vehicle routing problem that involve time windows. The aim is to distribute the slack time over a route to keep the vehicle waiting in promising locations where new requests are likely to appear. The focus is on the scheduling aspect by determining the arrival and departure time for each stop at a planned route. Mitrovic-Minic and Laporte (2004) have proposed waiting strategies for PDPTW instances. They proposed two general heuristics that are based on partitioning of the route. Branke et al. (2005) have formulated the waiting drivers problem and analytically derived an optimal waiting strategy for the single vehicle case and empirically tested various strategies for the multiple vehicle case. The first approach to utilize probabilistic knowledge in waiting strategies was proposed by Ichoua et al. (2006).

Relocation strategies aim at proactively positioning a vehicle in promising high-intensity regions. van Hemert and La Poutré (2004) derive fruitful regions from probabilistic knowledge and perform relocations of vehicles to anticipated requests. They have shown that the number of accepted requests can be maximized on PDP benchmark instances. Also in the area of emergency services relocation plays an important role as Pillac et al. (2012a) note. Work on the Emergency Vehicle Dispatching problem includes Gendreau et al. (2001) and Haghani and Yang (2007).

There are also some general strategies that aim at increasing the flexibility in terms of incorporating future requests. Batching strategies use request buffering to delay the assignment of non-urgent requests which has been investigated by Pureza and Laporte (2008). That way, the reaction time for urgent requests can be reduced. The double horizon approach proposed by Mitrović-Minić et al. (2004) is based on a modification of the evaluation

function. The objective for planning long-term requests is modified in such a way, that large slack times are favored to account for possible future requests. Diversion allows plan changes while the vehicle is moving to a location leading to an increased flexibility. It has been investigated by Ichoua et al. (2000).

2.2.4 Parallelization

The need for a fast reaction time in combination with the availability of multi-core processors and graphics processing units motivate the development of parallel metaheuristics for dynamic vehicle routing problems. Additionally, robustness is an important criterion and evolving multiple compatible plans / scenarios has shown to be beneficial leading to computationally intensive approaches that require the solution of multiple scenarios concurrently.

There are several strategies for the parallelization depending on the problem structure and the hardware architecture. Ghiani et al. (2003) have identified that important factors that have to be taken into account in the context of dynamic vehicle routing are the computational effort to generate a feasible and near-optimal solution as the situation changes, the available computational power, and the average inter-arrival time of dynamic information. In terms of hardware architecture, parallel architectures are now generally available by means of multi-core processors on desktop computers while coarse-grained architectures result by connecting general purpose PCs and workstations as mentioned by Ghiani et al. (2003).

Based on these considerations, a suitable parallelization strategy has to be derived. Since most of the research on solution strategies for dynamic vehicle routing focuses on heuristic approaches, the emphasis will be laid on parallelization strategies for metaheuristics. Crainic and Toulouse (2003) distinguished between three types in the case of metaheuristics:

- Type 1: This parallelization strategy is based on low-level parallism. The emphasis is on speedup while the behavior of the algorithm is not changed.
- Type 2: The search space is partitioned leading to a reduction in size. Multiple explorations can be performed to cover multiple parts.
- Type 3: There are multiple concurrent explorations of the complete search space.

A refined taxonomy that was initially presented by Crainic et al. (1997) refines this categorization and also takes control and communication strategies into account (cf. Crainic and Toulouse (2003)). A categorization is

made according to three dimensions. The control cardinality indicates if the search is steered by a single master process or by several processes. The search control type is characterized according to information exchange and synchronization. In terms of search differentiation, the individual processes can start from the same or different solutions and can be based on the same or different search strategies.

In the following, two parallel tabu search algorithms will be presented as well as the parallel execution of the MSA/MPA framework in terms of the different parallelization strategies. As noted by Pillac et al. (2012a), little work has been carried out on parallel solution methods for dynamic vehicle routing problems despite the importance of the topic.

A parallel tabu search for dynamic VRP has been presented by Gendreau et al. (1999) which is based on the adaptive memory concept. First an initial set of solutions is created using a construction heuristc. Then, several independent tabu search threads improve a single solution and store the resulting route in an adaptive memory. Within a tabu search process a decomposition of the solution to a disjoint set of routes is performed which are optimized in parallel. After a fixed number of iterations, the optimized routes are stored in the adaptive memory. Then, each process constructs a new initial solution from the adaptive memory and repeats the process. Crainic and Toulouse (2003) classified this algorithm as a Type 3 cooperative parallelization strategy that additionally applies decomposition within each process.

A different parallel tabu search was proposed by Attanasio et al. (2004) for the dynamic DARP. It can also be categorized as a Type 3 parallelization, however a different communication and control strategy is used. Multiple tabu search processes with different parametrizations are applied on the same search space. Some are parameterized for intensification while others are for diversification. Whenver a new global best solution is found, it is communicated among all search processes which are restarted with that solution. The algorithm has been tested with different number of threads and the benefits of parallelization in terms of solution quality have been shown.

An obvious low-level (Type 1) parallelization strategy for the MPA / MPS approach (Hentenryck and Bent, 2009) is to optimize the individual plans / scenarios on a parallel computing infrastructure. This is especially suited for parallelization, since they are optimized independently and the optimization of many different plans / scneario leads to a computationally expensive method.

2.3 Technical Requirements and Practical Implementation

The previous chapter focused on the algorithmic strategies for dynamic vehicle routing. In the following, a brief overview will be provided how these methods can be implemented for pratical applications. The necessary technical components will be outlined and decision support systems that have been implemented will be surveyed.

2.3.1 Technical Components

There are several essential technologies to provide operative decision support in dynamic vehicle routing environments which are linked together to provide real-time information processing. In that context, telematics systems combine telecommunications and informatics and integrate data acquisition, transmission, storage, and processing (cf. Richter (2005)).

A general architecture for real-time vehicle management systems is proposed by Giaglis et al. (2004). It consists of a back-end system for the dispatching center and a front-end system for the vehicles and drivers which are linked together by a communication element. In the dispatching center, route plans are created according to dynamically arriving information and communicated to the vehicles and drivers which in turn report back their state. This basic information flow can be implemented utilizing various technologies combined in an integrated telematics solution.

Larsen et al. (2008) note that the essential technologies required for implementing a dynamic vehicle routing system are positioning equipment, communication equipment and geographical information systems.

The positioning of vehicles in combination with frequent transmission of their location provides the dispatching center with valuable information about the current vehicle status. Different technologies can be utilized such as the Global Positioning System (GPS), triangulation by means of the mobile phone network, or manual determination by the driver.

The communication equipment links the dispatching centers with the vehicles. In that context, mobile communication systems can be used as well as radio-based and satellite-based systems. Larsen et al. (2008) point out that mobile communication systems are characterized by a high flexibility, low installation costs, and high operational costs. Alternatively, radio-based systems have higher setup costs but lower operational costs. Satellite-based systems can be integrated with other inter-modal communication systems (cf. Zaepfel and Vogl (2010)). In terms of communication with the driver,

on-vehicle board computers as well as integration with navigation devices can be used.

Geographic information systems (GIS) provide data about road-networks. In combination with real-time traffic information, they can be used to dynamically calculate shortest paths for the vehicles. Additionally, the visualization of the current situation within decision support systems can be achieved by means of digital maps to keep track of vehicle positions and status.

2.3.2 Decision Support Systems

A decision support system (DSS) in the area of transportation is "an interactive, computer-based system the supports the decision maker in solving a complex, usually unstructured (or poorly structured) transportation decision problem" (Zak, 2010). In the context of dynamic vehicle routing, the DSS is embedded in a changing environment receiving both static and dynamic data. The integration with other systems such as enterprise resource planning (ERP), GIS, communication, or traffic systems is crucial to receive data about the current state and communicate decisions to the drivers as well as customers (cf. Giaglis et al. (2004)).

Frequent monitoring of the current state such as newly arriving orders, vehicle positions, or traffic information is required as well as the incorporation of real-time routing and scheduling algorithms and the communication with drivers and customers. From these requirements, Pillac et al. (2012b) derive practical requirements for DSS. It should be event-driven to periodically update the state, parallelized to account for a short reaction time, and flexible due to the large number of variants.

Pillac et al. (2012b) surveyed several DSS for dynamic vehicle routing. Especially city logistics is a frequent application area. Attanasio et al. (2007) as well as Fleischmann et al. (2004) considered DSS for local area courier services. Barceló et al. (2007) and Zeimpekis et al. (2007) considered the distribution of goods in urban areas. Bieding et al. (2009) presented a DSS for the delivery of newspapers while Li et al. (2007) considered waste collection.

Summarizing, Pillac et al. (2012b) note that there is a growing body of research on dynamic vehicle routing which led to innovative DSS, however also pointing out that maturity has not been reached and important research challenges remain. Also, there is a gap between available optimization techniques and those incorporated in actual DSS available on the market calling for flexible optimization approaches.

2.4 Research Directions

Research on dynamic vehicle routing problems has reached a certain maturity level and has been implemented in larger and medium size business as Larsen et al. (2008) note. In a recent survey Pillac et al. (2012a) point out, that in the last decade there has been a growing interest on the underlying models of dynamic routing environments and the research has focused on developing methodologies that address the inherent dynamism and uncertainty. However, important research questions remain open which are summarized in the following from surveys of Pillac et al. (2012a), Larsen et al. (2008), Jaillet and Wagner (2008), and Ichoua et al. (2007).

In general, there is a lack of a taxonomy of dynamic VRP variants and there are no reference benchmarks that are widely used as both Pillac et al. (2012a) and Ichoua et al. (2007) observed. A taxonomy of different variants, as several exist for static VRP, would stimulate the development of specialized methodologies. A general classification based on the objective and degree of dynamism has been proposed by Larsen et al. (2002) but a detailed taxonomy still has to be derived. From a theoretic perspective Jaillet and Wagner (2008) highlight the need for measures for the evaluation of algorithms that allow rejecting requests, the investigation of the value of varying degrees of information as well as the combination of online routing with game theory.

In terms of algorithmic approaches, Pillac et al. (2012a) as well as Ichoua et al. (2007) have identified that the literature on parallel algorithms is still scarce pointing out the relevancy of that topic. The need for parallel methods is triggered by the availability of parallel architectures in combination with the need of fast reaction time and robust algorithms.

The anticipation of future events has been identified as an important research direction but is still not fully investigated. Although several surveys such as Ghiani et al. (2003) and Ichoua et al. (2007) have already pointed out the significance of advanced knowledge, Pillac et al. (2012a) note that most approaches still do not incorporate stochastic information. In that context, Ichoua et al. (2007) have identified that methodology has to be developed to analyze data and filter meaningful patterns combined with decentralized data processing. Larsen et al. (2008) as well as Ichoua et al. (2007) note that robustness concerning events such as vehicle breakdowns that are less predictable is still an open research question.

In terms of modeling, in many application areas there is a need for rich models that capture the specific characteristics of the problem environment. Larsen et al. (2008) state that the integration of industry practices such as the use of bikes in congested areas is an important modeling aspect. Pillac

et al. (2012a) highlights the need for models that go beyond focusing on efficient routing. An example they mentioned was the definition of flexible time windows.

Summarizing, research has mainly focused on specialized methods that are adapted to a problem environment whose characteristics do not fundamentally change over time. The goal of this thesis is to overcome this limitation. The basis for the considerations is a generic modeling approach based on simulation optimization that allows the creation of rich models taking into account parallelization and the anticipation of future events. Two main research directions are followed in this thesis to create novel adaptive heuristic approaches for dynamic vehicle routing investigating algorithmic and practical aspects.

As already pointed out in several surveys such as Larsen et al. (2008) or Ichoua et al. (2007), there is a need for specialized solution methods that are tailored to consider the specific characteristics of the problem environment. In this context, the focus is on specialized policies that are generated automatically based on simulation models. The aim is to create a solution methodology that can be adapted constantly to the problem environment.

Ghiani et al. (2003) as well as Larsen et al. (2008) have highlighted that hybrid variants should be investigated. Ghiani et al. (2003) mentioned environments with changing degree of dynamism while Larsen et al. (2008) presented overnight courier mail services as a suitable application area. As a response to this research question, a new hybrid approach is presented that is based on the adaptive selection of an appropriate solution methodology as the problem environment is changing. The selection is made utilizing a portfolio of several specialized algorithms.

Ultimately, combining the automatic generation and adaption of specialized policies based on simulation models and a situational selection of a suitable solution method forms the methodological basis for decision support systems in the area of dynamic vehicle routing that are adaptive in terms of changing problem environment characteristics.

Chapter 3

Simulation-Based Optimization of Production and Logistic Scenarios

In this chapter, a generic simulation and optimization environment for considering archetypical as well as practical variants of dynamic vehicle routing problems is motivated and introduced based on a generic modeling framework. It is the basis for the consideration of algorithmic as well as practical aspects and the methodological developments within this thesis. To illustrate its applicability for practical applications, three concrete scenarios in the area of production and logistics are presented.

The remainder of this chapter is structured as following. In Section 3.1 the simulation and optimization environment is presented. Based on that environment, practical case studies are presented in the fields of steel production (Section 3.2), the production of firefighting vehicles (Section 3.3), and the distribution of groceries (Section 3.4).

3.1 Simulation Optimization Environment

For considering dynamic vehicle routing problems that include complex side constraints, applying discrete event simulation coupled with metaheuristic optimization algorithms is a commonly used approach (cf. Ghiani et al. (2003)). Practical problem formulations typically involve dynamic interactions as well as stochastic influence factors that are too complex to be treated analytically. Simulation optimization is a viable alternative to be applied for large-scale stochastic and dynamic systems (Fu et al., 2005).

In the context of dynamic vehicle routing, a simulator provides dynamic

updates in the form of events such as appearance of new orders or vehicle movements. It is coupled with an optimization algorithm that reacts to the changes and returns decisions that are executed by simulated vehicles. This corresponds to a control (dynamic) optimization approach where the optimization is an integrative part of the simulation environment as opposed to a parameter (static) optimization where certain parameters of the simulation are optimized (cf. Gosavi (2003)).

A simulation optimization approach is especially useful for developing and testing algorithms since it would be difficult to evaluate the algorithm performance in a real system on a large number of cases (cf. Fleischmann et al. (2004)). Additionally, a simulation environment can be used for a sensitivity analysis to provide decision support for tactical or strategical decisions by considering different (possibly fictional) scenarios and evaluating their optimization potential.

A generic and extensible modeling infrastructure is an important basis for the developments presented in this thesis. In terms of problem modeling, several different problem variants are investigated. Pillac et al. (2012b) motivates the importance of flexibility in that context due to the different application areas of dynamic vehicle routing. In terms of the proposed algorithmic strategies, it is required to evaluate and combine different methods within a single modeling infrastructure and apply them to different problem variants.

Few generic approaches have been investigated and research has mainly focused on specialized decision support systems tailored to a particular application areas such as city logistics (Barceló et al., 2007). One of the few approaches to create a generic simulation and optimization framework for dynamic vehicle routing was presented by Pillac et al. (2012b). While that framework is generic in terms of application area it is not in terms of methodology since it focuses on the multiple scenario solution approach.

In the context of this thesis, a framework for the simulation and optimization environment is required that is both generic in terms of problem variant as well as solution methodology. It should be easily extensible by means of modules to support additional problem variants and algorithmic strategies.

Considering these requirements, a modeling framework for the simulation and optimization of dynamic vehicle routing problems is proposed. The framework is integrated in the optimization environment HeuristicLab (Wagner et al., 2014) which is a suitable basis for the implementation since it is flexible and extensible by means of plugins and provides different algorithms and models for vehicle routing.

HeuristicLab has already been coupled with simulation environments in various simulation optimization approaches. Beham et al. (2008) as well as

Beham et al. (2012) outlined the coupling of HeuristicLab with simulation environments based on generic interfaces and Beham et al. (2009a) studied the application and behavior of evolutionary algorithms. Practical case studies include solving buffer allocation (Can et al., 2008), facility layout (Beham et al., 2009c), scheduling (Pitzer et al., 2011), and transportation (Vonolfen et al., 2013a) problems.

The integration in HeuristicLab allows a flexible interfacing between the simulation and optimization components. For instance, algorithms available in HeuristicLab can be integrated in simulations to dynamically make decisions (control optimization) while optimization algorithms can be applied to parametrize simulations (parameter optimization).

3.1.1 Generic Simulation Optimization Core

The basis of the modeling approach is a generic core from which specialized models for different problem variants are derived. Two concrete specializations for dynamic pickup and delivery problems as well as inventory routing problems are presented which are the basis for the further considerations in this thesis. The core consists of a modeling framework for deriving specialized simulation and optimization environments. It defines basic interactions and interrelations between abstract elements that are concretized for particular application areas.

The modeling approach has evolved over time and preliminary results have been published previously. In Vonolfen et al. (2010), a generic system architecture was presented coupling an external simulation environment with HeuristicLab. Later, the simulation environment has been integrated in HeuristicLab directly to allow a more flexible interfacing. The decoupling of simulation and optimization by means of an event mechanism with a special emphasis on the incorporation of static algorithms was presented by Vonolfen et al. (2012a). In the following, a generic modeling approach is derived based on these previous developments.

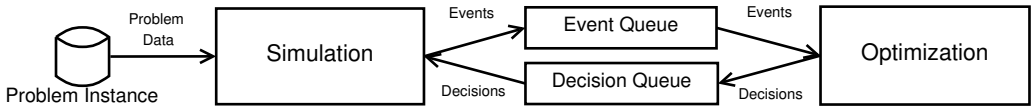

Figure 3.1: Basic architecture and information flow of the simulation optimization approach

The basic architecture and information flow is depicted in Figure 3.1. The simulation and optimization component exchange information by means

of events that represent changes in the simulation model and decisions that are derived from the current state. The simulation model has access to problem data such as predefined events or probability distributions in the form of a predefined problem instance to be able to reproduce a particular simulation run and test various optimization approaches. The simulation and optimization model are decoupled by an event queue and a decision queue.

The simulation progresses in discrete time steps. A time step can represent an arbitrary time span such as a second or a minute allowing to control the simulation speed in relation to the real-world. At each step, external events are triggered as well as events resulting from executed decisions. External events such as appearance of new orders or time-dependent travel times are retrieved from an external problem instance. The decisions are retrieved from the optimization component and trigger resulting events when they are executed such as vehicles moving or orders being served.

The simulation and optimization component are decoupled by a clearly defined communication protocol to ease the integration of the optimization approach in a real decision-support system. The decoupling by means of queues allows both synchronous and asynchronous communication to be realized. While real decision support systems are intrinsically asynchronous, in testing environments often synchronous communication between the simulation and optimization component is used. This eases implementation and is a feasible assumption in cases where reaction time is not of crucial importance or the computation time of the optimization algorithm is set to a fixed time slice.

The simulation and optimization core is designed in a flexible way to allow both synchronous and asynchronous communication. This is achieved by creating separate execution threads for the simulation and optimization component. In the case of synchronous communication, the simulation waits for processing the next time step until the optimization algorithm has finished calculations for the current step. In contrast, the simulation time progresses without considering the state of the optimization component while filling the event queue when asynchronous communication is used.

The basic modeling concept revolves around the two communication elements. In the simulation, events are triggered and decisions are executed while the optimization reacts to events and provides decisions for the current situation. This scheme is provided as a basic template by means of abstract elements that are specialized for particular application areas. In Figure 3.2, the basic template is illustrated and the abstract elements are depicted in italic style while the concrete elements are in bold.

For each application area, specialized events such as the occurrence of a new request or a vehicle movement as well as specialized decisions such

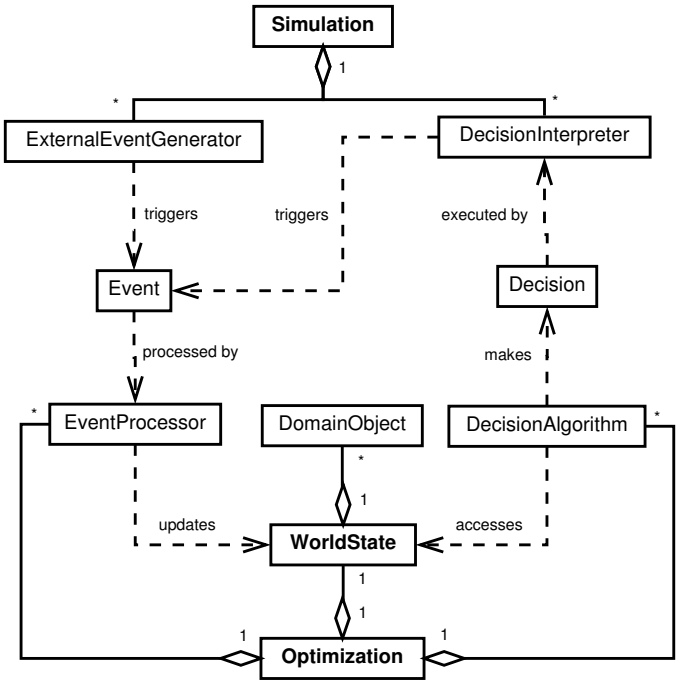

Figure 3.2: Template of the generic simulation and optimization core

as serving a customer or relocating a vehicle to a certain location are modeled. The proposed template revolves around these two basic communication elements.

The simulation component consists of multiple *ExternalEventGenerator* as well as *DecisionInterpreter* elements. External events are generated by *ExternalEventGenerator* elements that may access problem data such as predefined events or probability distributions. *DecisionInterpreter* elements execute decisions within the simulation by triggering resulting events such as vehicle movements or served orders.

In the optimization component, changes in the forms of events are handled by *EventProcessor* elements. For each specialized event, an associated processor must exist. The event processors update the *WorldState* which captures the current state of all relevant *DomainObject* elements such as vehicles or orders. A *DecisionAlgorithm* accesses the current state and makes decisions that are in turn executed by the simulation.

Due to the large variety of different solution methods for dynamic vehicle routing problems, a special emphasis is on providing a generic template for decision algorithms which is depicted in Figure 3.3.

Both event-based dispatching policies as well as static algorithms can be incorporated in the framework. Dispatching policies can access the world

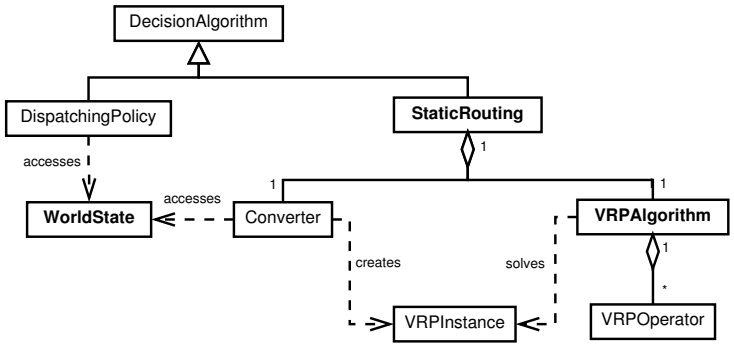

Figure 3.3: Template for decision algorithms in the optimization component

state directly and make decisions accordingly. For the application of static routing algorithms (*VRPAlgorithm*), the world state is transformed in a static VRP instance (*VRPInstance*) by a wrapper element (*Converter*).

The static instance can be solved by one of the various routing algorithms available in HeuristicLab. The underlying flexible operatior model of HeuristicLab (cf. Wagner et al. (2014)) allows a re-use of algorithmic building blocks for different VRP variants by connection problem properties with operator capabilities. Details can be found in Vonolfen et al. (2012a). A special emphasis on evaluating the integration of different static algorithms using HeuristicLab was laid in the work of Vonolfen et al. (2010) as well as Vonolfen et al. (2012a).

Summarizing, a flexible core framework is provided within HeuristicLab for modeling different VRP variants. Static routing algorithms available in HeuristicLab can be incorporated by transforming the world state into a static problem instances. Other solution strategies such as dispatching policies can be integrated directly.

3.1.2 Specializations

In order to create a simulation and optimization environment to investigate certain dynamic VRP variants, the presented abstract template is specialized. This is achieved by deriving concrete realizations from the abstract elements. In particular, the following elements are defined that are specific to the application area:

- Events: Changes in the world state are triggered by events. In the simulation model, corresponding event generators are implemented in in the simulation and event processors in the optimization component.

- Domain objects: The domain objects represent the current world state

held by the optimization component that is updated as new events are observed.

- Decisions: Actions taken during the planning process are represented by decisions. In the optimization component, specialized decision algorithms are implemented while in the simulation they are executed by corresponding decision interpreters.

- Decision Algorithm: Specialized solution methods can be implemented for each application area. In the case of the adaption of static algorithms, a converter is defined that transforms the current world state into a static VRP instance. Also specialized methods such as dispatching policies can be realized, that access the world state directly.

In the following, two concrete specializations of the simulation optimization environment will be presented. In particular, pickup and delivery problems as well as inventory routing problems will be considered. These two problem variants are investigated both from the algorithmic and the practical perspective based on the presented simulation optimization environment.

Pickup and delivery problems

Pickup and delivery problems (PDP) involve transport requests between geographic locations that are executed by a fleet of vehicles. Consequently, requests and vehicles are the two domain objects in this model. The events related to requests are the appearance of new requests and state changes of requests such as cancellations or service completions. In terms of vehicles, considered events are changes in their state such as movements, services, or breakdowns. Decisions in PDP involve planning the sequence the vehicles should service the requests or relocating them to / waiting at strategically good positions. The simulation optimization environment for PDP modeling these interactions is illustrated in Figure 3.4.

In terms of the decision algorithm, there are several solution strategies that can be considered. On the one hand, specialized dispatching policies can be derived and on the other hand static routing algorithms for PDP can be adapted. The latter requires a special converter that maps the world state to a multi-depot PDP instance. Each vehicle is modeled as a separate depot whose location is changed as the vehicle is moving. That way, standard algorithms for static PDP variants can be used. The static algorithm works with operators designed for PDP.

The simulation optimization environment for PDP variants will be used in several sections of the thesis. In Section 3.2, a PDP model for the op-

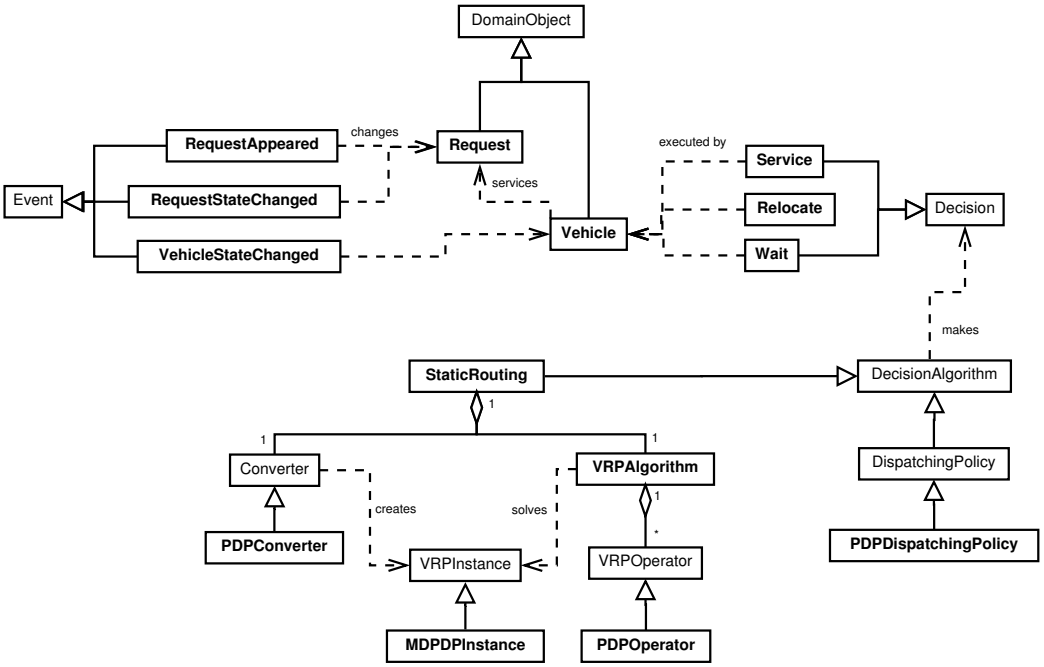

Figure 3.4: Specialized simulation optimization environment for pickup and delivery problems

timization of transport activities in the context of steel production will be considered. The standard PDP model is enrichted with specific constraints of that domain. In a similar manner, transport activities are optimized in the context of the production of firefighting vehicles in Section 3.3. Especially the integration with warehousing is investigated. Standard variants are considered for algorithm development. In Section 4.3 dispatching policies are generated for dial-a-ride benchmark instances, while in Section 4.4 waiting strategies are generated for PDP benchmark instances with time windows.

Inventory routing problems

Inventory routing problems combine inventory management with routing. The specialized simulation optimization environment is illustrated in Figure 3.5. The central domain objects are customers having a stock of goods. For each good a consumption rate, the current stock situation, and a storage capacity is given. The simulation uses a certain degree of abstraction not simulating the exact operation of the vehicles but using a whole planning period, which is usually a whole day of operation, as a time step.

In each planning period it is decided what goods to replenish and the route that determines the sequence in which the vehicles visit the customer.

The considered events are stock updates triggered by consumption of goods as well as replenishments. Since it is assumed that the exact consumption rate is unknown in advance the problem is considered dynamic.

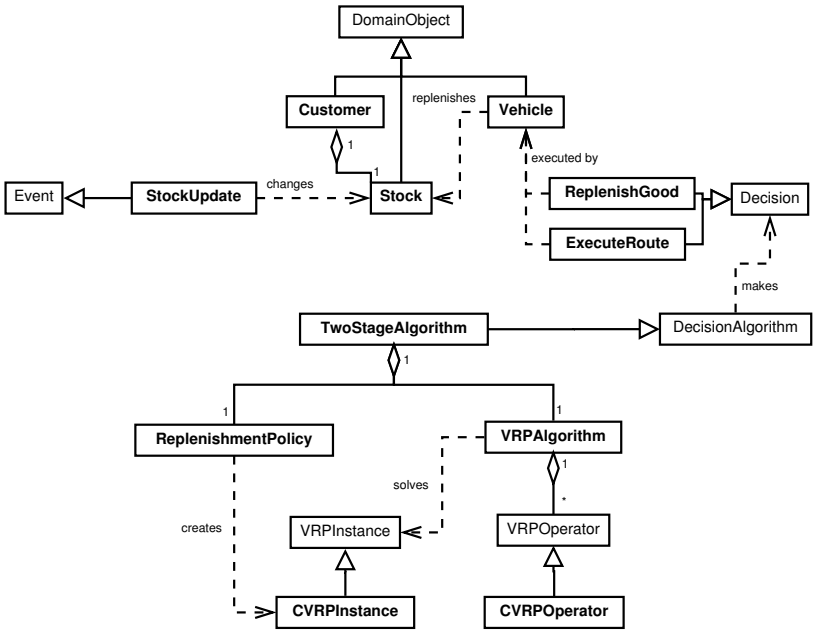

Figure 3.5: Specialized simulation optimization environment for inventory routing problems

In terms of decision algorithm, a two-stage approach is applied. In the first stage, a dispatching policy is applied to decide what goods to replenish and in the second stage the routing problem is solved. In this example, a CVRP model is used to represent the routing problem of a single day created by the replenishment decisions.

Consequently, after the planning phase concerning the replenished goods has been completed, a CVRP instance is derived which can be solved by a standard CVRP algorithm to derive the visiting sequence. Also other algorithmic strategies can be incorporated in that algorithmic framework. The two-stage approach has the obvious advantage that standard routing algorithms can be utilized.

The simulation optimization model for inventory routing problems will be used for considering benchmark instances derived from real-world data that involve stochastic consumption rates which are revealed after each day of operation. In particular, the distribution of goods in retailing will be considered. In Section 3.4 the modeling aspects of this practical variant will

be examined while in Section 4.2 the focus is on the algorithmic aspects in terms of simulation-based generation of replenishment policies.

In the following, three practical case-studies will be presented that are based on the simulation optimization environment and have been implemented in HeuristicLab.

3.2 Optimization of Transport Activities in Steel Production

In the steel production process, material handling in general and scheduling transport activities in particular are challenging tasks due to interdependencies with upstream and downstream processes and specific handling constraints. There are several operational constraints and dynamic interactions that would be difficult to represent by a purely static model.

The aim of the presented work is to algorithmically schedule and route transport activities within cold charge steel production using a rich problem formulation. A case study using data from an Austrian steel factory is presented that is based on a simulation model coupled with an optimization algorithm. The resulting schedules are compared with the original solutions created by human experts to evaluate optimization potentials and bottlenecks. Using a realistic model, the drawn conclusions can be transferred back into practice to improve the material handling process of steel slabs.

This section is based on a case-study dealing with a static problem formulation previously published by Vonolfen et al. (2013b). Paraphrased parts include the simulation model, optimization algorithm, and results of optimizing the static model. The main contribution within the scope of this thesis is the consideration of a dynamic problem variant and an extended interpretation and analysis of the optimization potentials.

3.2.1 Context and Motivation

The focus of this work is the optimization of scheduling and routing transport activities within the cold charge process to identify optimization potentials and bottlenecks. The case study is based on the scenario and data from an Austrian steel plant. A detailed simulation model is presented to capture important dynamic characteristics of the process. The simulation model is coupled with a routing algorithm to derive an optimized transport schedule. The optimized schedules are analyzed to draw conclusions about bottlenecks and optimization potentials in the material handling process of steel slabs in cold charge.

The process of steel production is generally complex, multistage, geographically distributed and energy as well as capital intensive. As a result, the involved material handling activities have to be managed as efficiently as possible. Starting with raw materials such as iron ore and coal, the melted steel produced in the furnace is transformed into slabs at the continuous casting machine.

An important activity is the transport and storage of these slabs which are intermediates in steel production and are later transformed by the rolling mill into products such as plates or coils. The material handling is an energy-intensive activity since slabs weigh approximately 25 tons and are around five to twelve meters long.

A general distinction can be made between the cold charge and hot charge process. In the hot charge rolling process, slabs are transported from continuous casting to the rolling mill without any additional storage. Since it is very energy consuming to reheat cooled down slabs, the aim is to preserve as much energy as possible and keeping the slabs at a high temperature. As a result, casting and rolling are strongly coupled and synchronized in the hot charge.

On the contrary, casting and rolling are decoupled by an intermediate storage in the cold charge process. The slabs are stored in a slab-yard after casting and picked and transported to the rolling mill afterward. An obvious disadvantage of the cold charge is, that the slabs cool down at the storage and have to be re-heated afterward. However, due to technical limitations, modern steel factory have a hot charge ratio of about 60% making it important to manage the overall material handling process as efficiently as possible.

There are several studies dealing with scheduling of the main production activities, which are steel making, continuous casting, and rolling as surveyed by Tang et al. (2001a). A main research challenge is the integration and coordination of the different activities. Relatively few studies have investigated material handling activities, which are the storage and transportation of steel slabs.

In terms of storage, both the storage assignment and efficient retrieval of slabs have been investigated. The assignment of storage spaces for slabs in the cold charge process has been investigated by Kofler et al. (2012) using a storage assignment model and local search. A general model for the stacking of slabs has been proposed by König et al. (2007) and tests have been carried out on real-world instances. The slab stack shuffling problem, which has been defined by Tang et al. (2001b), deals with selecting steel slabs for creating a rolling schedule in such a way that the number of shuffles required to retrieve them is minimized. Dynamic programming (Tang and Ren, 2010) as well as genetic algorithms (Tang et al., 2002) have been applied.

The transportation of slabs using cranes and the generation of schedules using a two-stage heuristic approach was considered by Dohn and Clausen (2010). The results have been verified using simulation. In a larger context, Zäpfel and Wasner (2006) studied the storage and transport of steel coils. Related problems occur especially in the area of container terminals, since there the handling of bulky items is also an important aspect. The routing of straddle carriers was investigated by Kim and Kim (1999), modeled as a carrier routing problem and solved using beam search. A dynamic routing approach for yard trailers was investigated by Nishimura et al. (2005) where they could achieve container handling cost savings.

In this work, the focus is on routing and scheduling the transport activities in the cold charge process. The typical life-cycle of a slab in cold charge is illustrated in Figure 3.6. After the slab is created at the continuous caster, it is stored in a slab yard. Some slabs have to be post-processed at aggregates such as scarfing or cutting for quality assurance reasons. In the case of the cold charge process, casting and rolling are decoupled via the slab yard. Some slabs are stored there for a long period of time before they are scheduled for rolling.

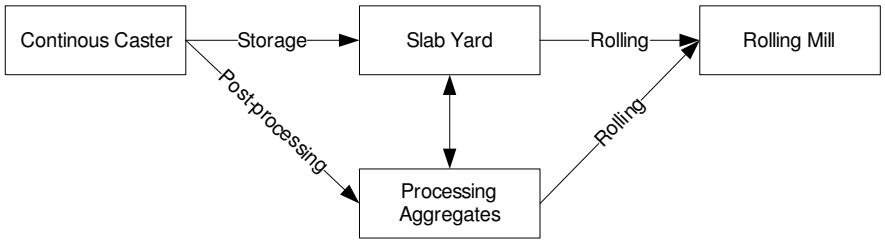

Figure 3.6: Typical life-cycle of a slab in the cold charge steel production process (cf. Vonolfen et al. (2013b))

Straddle carriers transport the slabs between the continuous caster, the slab yard, the processing aggregates and the rolling mill. They also maneuver in the slab yards and store and pick the slabs there. They can carry up to 105 tons which are usually around 4 slabs. At the continuous caster, the processing aggregates, and the rolling mill special handover zones exist where the slabs can be picked up by the straddle carriers.

The outside slab yard is organized in several fields which contain several rows of slab stacks. The storage position is determined by an external storage assignment procedure which has been investigated by Kofler et al. (2012). Some slabs are stored for a duration of several weeks until they are further processed. When a slab has to be retrieved in the slab yard that is stored beneath other slabs, shuffling operations have to be performed.

It is a challenging task to sequence the transport activities in an efficient way which is mainly carried out by a human expert. Many operational constraints have to be considered since the transport links all related activities together. To evaluate the optimization potential concerning the material handling of the cold charge slabs a simulation optimization approach is followed. The goal is to minimize the total lead time and thus increasing the throughput by efficient routing and scheduling.

The operational constraints make it necessary to consider dynamic effects and inter-dependencies between the individual straddle carriers and also between transport and other activities. As a result, the system is modeled using discrete-event simulation to achieve as accurate results as possible. Simulation-based optimization is applied to sequence the transport activities in such a way that the throughput is increased.

3.2.2 Simulation Model

A detailed simulation model is created to capture all important characteristics of the process that would be hard to model using a closed form. Consequently, a model of the system is needed that captures the important characteristics of the operations to achieve practically comparable and implementable results. The transport activities basically involve pickup and delivery operations. As a result, the simulation model is based on the pickup and delivery model presented in Section 3.1.2 and extended with the specific constraints.

There are three different kinds of transport activities. After being cast slabs have to be transported from the continuous caster to the slab yard for storage. Directly after casting or during storage, some slabs have to be transported to processing aggregates for quality assurance reasons. Finally, slabs are transported from the slab yard to the rolling mill for being processed in a rolling schedule. The rolling schedule is known at the beginning of a shift while the other transport requests arrive dynamically.

The simulation is carried out in discrete time steps each representing one minute in real time. Three different types of events are carried out where the simulation state is updated:

- The straddle carriers perform actions which are picking up, transporting and delivering slabs according to a given schedule. The original schedule was created by a human expert and can be compared with schedules created by an optimization algorithm.
- The handover places are updated by simulated crane movements which are performed exactly as in the original shift.

- The rolling mill state is updated according to the rolling schedule which is known beforehand.

The transport requests have to be routed and scheduled according to several operational constraints which are be categorized into shuffling, rolling, temporal and capacity constraints.

The shuffling constraints concern pickup and delivery operations originating from slab stacks at handover places or at the slab yard:

S1 No shuffling can be performed at handover places. The slabs have to be transported in the exact sequence they are stacked. In the slab yard shuffling is possible by relocating slabs to other stacks.

S2 Only one straddle carrier can operate on an individual handover place or in a lane at a slab yard at the same time due to security reasons.

A stock model consisting of all handover places and the outside slab yard is the basis for checking the shuffling constraints. It holds the current storage position of each slab and is updated whenever a slab is moved by a straddle carrier or cranes. The crane movements are considered to be fixed and lie outside the system scope while the straddle carrier movements are subject to optimization. Only slabs located on top of stacks can be retrieved on handover places while in the slab yard shuffling operations can be performed (S1). For security reasons, only a single straddle carrier can operate at once in a given lane in the slab yard or at a given handover place which is ensured by locking (S2).

The correct transport of slabs to the rolling mill is ensured by rolling constraints considering the rolling mill schedule:

R1 A certain security buffer of slabs cannot be under-run at the rolling mill. A certain number of slabs that are due to be rolled next have to be available at the rolling mill at any time resulting in implicit time windows where slabs have to arrive at the rolling mill.

R2 The sequence the slabs arrive at the rolling mill has to follow the scheduled rolling sequence to some degree. Cranes at the rolling mill can still re-shuffle incorrectly delivered slabs, however these shuffling operations are limited.

To check to buffer (R1) and shuffling constraints (R2) at the rolling mill, its rolling schedule is simulated by removing the slabs from the buffer the time they have been processed and considering the slabs that have been delivered.

The availability and due times of slabs and straddle carriers is determined by temporal constraints:

T1 There are scheduled maintenance tasks and driver breaks during shifts which have to be considered.

T2 For slabs that have to be processed, time windows exist at which the have to arrive at the processing facility the latest. For slabs at handover places, the time they are on top of a stack is implicitly determined by the predefined crane movements.

Straddle carriers can carry multiple slabs. However there are capacity constraints derived from their physical properties:

C1 Each straddle carrier can carry up to 105 tons.

C2 The difference in the dimension and size of slabs that are transported together cannot exceed certain limits.

The temporal (T1, T2) and capacity (C1, C2) constraints are checked for each action performed in the schedule of a straddle carrier.

All constraint violations are recorded during a simulation run as well as the travel and service effort of each straddle carrier. These results of the simulation run are used to compare transport routes and schedules created by human experts and optimization approaches.

3.2.3 Optimization Approach

The optimization algorithm aims routing and scheduling the transport activities of the straddle carriers in such a way, that the throughput of the transport system is increased while considering all operational constraints. During a simulation run, different key values are collected which are combined to a single quality value to evaluate a given plan. The evaluation function is given in Equation 3.1 (cf. Vonolfen et al. (2013b)).

$$\begin{aligned} \min(&travelTime + serviceTime \\ &+ \alpha * shuffling Violations \\ &+ \beta * rolling Violations \\ &+ \gamma * temporal Violations \\ &+ \delta * capacity Violations \end{aligned} \tag{3.1}$$

The main objective is the minimization of the *travelTime* and *serviceTime* of the straddle carriers and thus increasing the throughput. While the travel effort is the time the vehicles are moving between different locations, the

service effort is the time required for picking up and delivering slabs. In the case of handover places no shuffling is possible so the service effort of retrieving and delivering slabs consists only of the time required for a single lift. When retrieving slabs from the outside yard, shuffling operations are required if the slabs are located beneath other slabs in a stack. Furthermore, the creation of temporary stacks is required when slabs are retrieved from multiple rows of a single yard or from multiple yards. In that case, the slabs picked up at a given row are put on a temporary stack which is moved to each row where slabs are picked up.

The operational constraints are considered by adding a penalty to the quality of a solution if any constraints are violated. The penalty weights α, β, γ and δ are adapted during the search process. If the current solution is feasible, the penalty is decreased while it is increased if the constraint is violated. This methodology is detailed by Cordeau et al. (2001) who propose an exponential adaption of the penalty weights.

For optimization of the transport schedule, a tabu search heuristic is applied that works on an extended pickup and delivery problem formulation. As a starting solution the schedule that has been created by the domain expert is used. The algorithm then iteratively improves this schedule. The unified tabu search algorithm was proposed by Cordeau et al. (2001) and adapted to the pickup and delivery problem by Cordeau and Laporte (2003). It is based on iteratively exploring the neighborhood while preventing cycling by using a tabu list.

In the context of this case study, the neighborhood structure is based on the removal of an activity from the schedule of a straddle carrier. After removal, the activity is inserted in the schedule of another carrier at the best position. Moving back the activity to the original carrier is made tabu for t iterations. During an iteration, all possible insertion operations that are not tabu are evaluated and the best is applied.

Evaluating different insertion positions is a runtime intensive operation if a full simulation has to be performed each time. As a result, a more efficient procedure is needed for neighborhood exploration. A compromise has to be found between runtime and accuracy. For that purpose, a mixed evaluation is proposed that combines a static with simulation evaluation. For evaluating a given insertion position, a static evaluation is applied that does not consider all constraints but serves as a lower bound. Promising insertion positions are evaluated with a full simulation run.

The static evaluation is a lower bound for the quality since it is a relaxed formulation of the full simulation model. In the static evaluation, the detailed stacking operations and the interactions between the carriers and cranes are not considered. These include the exact service times, the locking of fields

and handover places and the crane movements. These aspects result from dynamic behavior that is validated in the full simulation. A full simulation is then only applied to the i insertion positions with the best evaluation where the one with the best quality value after the full simulation is chosen.

To determine the number i of insertion positions that should be evaluated by simulation during the neighborhood exploration, preliminary experiments have been carried out on exported shifts. More details about the experimental setup can be found in Section 6.1.

	Static Speedup (factor)	Static Gap (delta)
Shift A	18.34	33.40%
Shift B	10.65	11.49%
Shift C	13.25	19.13%
Shift D	13.36	14.94%
Shift E	13.45	14.99%
Shift F	13.55	14.86%
Shift G	13.25	15.35%
Shift H	31.81	17.97%
Shift I	14.05	14.86%
Average	14.17	17.44%

Table 3.1: Comparison of the static with the simulation evaluation (cf. Vonolfen et al. (2013b))

A comparison between the simulation and static evaluation in terms of accuracy and runtime is provided in Table 3.1. The runtime and accuracy of the static and simulation evaluation has been performed on the initial schedule created by the human expert to gain insight about the parametrization. On average, the runtime of the simulation is about 14 times longer than the runtime of the static evaluation. The loss of accuracy is about 17% on average and varies from shift to shift.

Since there is a loss of accuracy in the static evaluation, a compromise has to be found between the chosen sample size i and the required runtime. For determining the value of parameter i preliminary experiments have been carried out with different parameter settings which are summarized in Table 3.2. A single optimization run has been carried out on Shift A with a fixed runtime of 20 hours.

The results show, that setting $i = 10$ provides the best tradeoff between accuracy and evaluations that can be performed. It allows to compute about 67,000 simulations, about 1,000,000 static evaluations and executing 25 iter-

	Iterations (count)	Simulations (count)	Static Eval. (count)	Quality (delta)
1 Sample	35	9,380	2,186,433	+2.24%
10 Samples	25	67,000	1,130,067	(reference)
20 Samples	17	91,120	722,885	+1.30%

Table 3.2: Comparison of different sample sizes by performing an optimization run on a shift (cf. Vonolfen et al. (2013b))

ations during the fixed runtime which achieves the best solution quality.

After the initial experiments and based on the parametrization proposed by Cordeau et al. (2001), the parameters of the tabu search have been set. The number of iterations n are set to 25 and for each insertion operation, the $i = 10$ best positions are evaluated with a detailed simulation. The tabu tenure t is set to $5 * log_{10} * |A|$, where $|A|$ is the number of requests.

Optimization runs are carried out both on static and dynamic problem formations to analyze the influence of the degree of dynamism. In the static problem variant, all information is known beforehand and at the beginning of the shift a schedule is created. In the dynamic problem variant it is assumed that the rolling schedule is known in advance while the other requests arrive one hour before they have been carried out in the original schedule. A re-optimization of the current schedule occurs each hour considering the newly arrived requests. Apart from the dynamically arriving storage and processing transport requests all other information is considered to be known beforehand.

3.2.4 Conclusions

Based on the experimental results presented in Section 6.1 several conclusions can be drawn about bottlenecks and optimization potentials in material handling within the cold charge steel production process. The optimized schedule is compared to the original schedule created by the human expert during the shift.

In total, the optimized schedule increases the throughput of the transport system by reducing the travel effort by 3.33% and the service effort by 6.01% leading to a total saving of 4.48%.

In terms of travel effort, even though the total number of trips is increased by 4.17% in the optimized schedule, the savings in terms of travel effort results from reducing the time the vehicles travel empty by 10.68%. This indicates, that the long term planning horizon of the algorithmic approach leads to a better trip sequence compared to the original schedule.

The main potential for reducing the service effort lies in the stacking and shuffling operations performed in the slab yard. In the optimized schedule, the shuffling effort is reduced by 5.44% which is achieved by scheduling the transport activities in such a way that slabs that are on top of stacks are transported before slabs that are at the bottom reducing the number of shuffles required. However, it must be noted that for some shifts this effort is not reduced due to the long period of time slabs are stored at the yard which means that some shuffles cannot be avoided during a single shift. On the other hand, the effort for creating temporary stacks when collecting slabs from multiple stacks can be reduced significantly for all shifts. In total it is reduced by 30.65% which mainly results from performing more trips to the storage and as a result avoiding the creation of temporary stacks.

The shuffling and stacking activities performed in the slab yard represent a large part of the total service effort. The algorithmic solution uses a quite creative way of working around this problem. By performing more trips to the storage area, stacking and shuffling operations are avoided. This reduction of capacity utilization is compensated by minimizing the time traveled empty by sequencing the individual trips more efficiently.

For a fair comparison between the original and the optimized it must be noted that the optimized schedule is created a-posteriori while the original schedule was created online during the shift by the human expert. When considering a scenario with dynamically arriving information, the trips are not sequenced as efficiently by the algorithm as in the case when all information is known in advance. The service effort can still be reduced by 3.59%, however the travel effort is not. In total, no statistically significant reduction in total effort can be detected in the case of the dynamic problem formulation.

The effects of dynamically arriving information on the solution quality should be still investigated in more detail in future work by exporting additional shifts. For some shifts, even an increase in the total effort can be observed in the optimized schedule compared to the original schedule. This indicates, that the human expert had more information than was available to the optimization algorithm. To mitigate this fact, an online stochastic model should be investigated that includes a forecast about future events.

In general, the main bottleneck lies in the stacking and shuffling operations in the slab yard. To reduce this effort, an efficient storage assignment is needed that considers the rolling schedule. A first effort to optimize the storage assignment in the slab yard was done by Kofler et al. (2012). A main challenge in this context is that storage and rolling are decoupled which could be mitigated using a forecast model for the rolling schedule. In the long term, an integrated model should be created that combines the optimization of casting, transport, storage, and rolling considering the inter-dependencies.

3.3 Integrated Warehousing and In-House Transport

In manufacturing environments, material handling is a crucial activity that accounts for up to 20% - 50% of the total costs as noted by Tompkins (2010). Material handling typically consists of several sub-activities such as storing and picking in the warehouse as well as in-house transport to the workstations. These activities are interconnected and their interrelations have to be considered when optimizing the overall material handling process.

In this work, the in-house transport is modeled as a dynamic pickup and delivery problem and integrated with warehouse picking and storing to an overall model. The dynamic routing model is not considered individually but linked to the down-stream warehouse activities to account for the dynamic interactions between warehousing and in-house transport. By coupling the dynamic routing model with warehousing optimization and simulation models, the interactions between these activities are investigated.

This section is based on a previously published study of Vonolfen et al. (2012b) from which several parts are paraphrased. The main contribution in the scope of this thesis is the creation of a simulation and optimization model for the in-house transport and its coupling with a warehousing simulation and a storage assignment optimization model. The warehousing simulation model was investigated by Kofler et al. (2010) which was implemented in the simulation software AnyLogic©. The storage assignment optimization approach was previously published by Kofler et al. (2011) as well as Kofler et al. (2014).

3.3.1 Context and Motivation

Typical production processes incorporate several interrelated activities that influence each other. This observation motivates the investigation of integrated problem formulations that combine several models and algorithms that deal with different aspects of the production process. In the context of this thesis, especially the integration of dynamic vehicle routing with other down- and upstream activities in production processes is relevant.

To illustrate the potential of a holistic approach, a study is carried out that is based on data from a practical scenario at a production site of one of the largest suppliers of firefighting vehicles and equipment in the world. In the context of the overall material handling process, the focus is on the investigation of interconnections and interrelations between warehouse picking and forklift routing.

A schematic illustration of the considered production site is depicted in Figure 3.7. The considered scenario includes a high-rack storage area which is decoupled from the production line by means of an intermediate handover zone. More than 10.000 parts are stored in the high-rack storage which are commissioned to the handover zone by six workers and two forklifts. They are then further transported to the workstations by four smaller forklifts which are operated at the workstation area. Additionally, work-in process parts are also transported between the workstations and back to the handover zone to be stored in the warehouse.

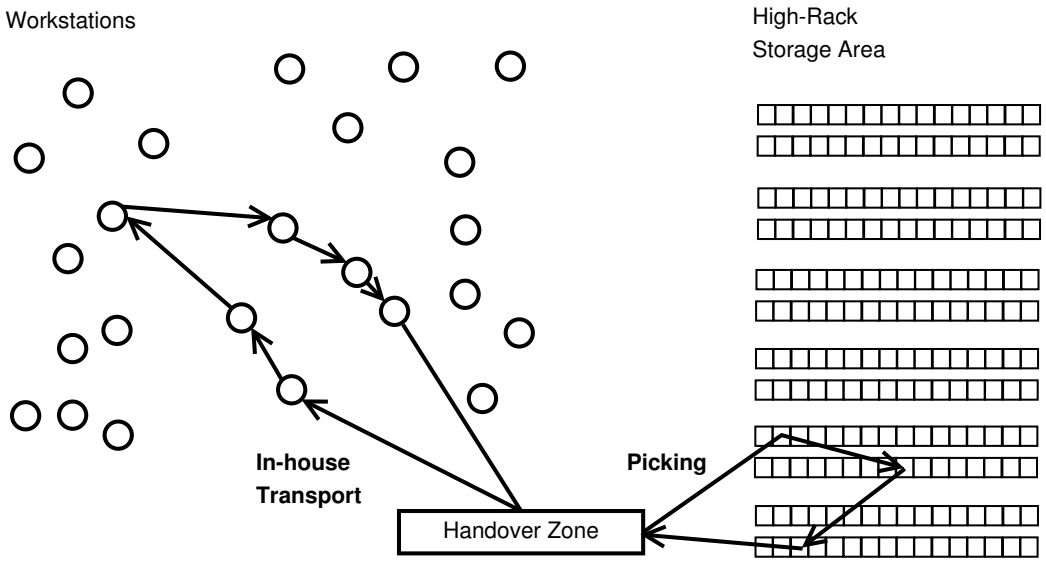

Figure 3.7: Schematic illustration of the production site layout showing an exemplary picking of materials from multiple storage spaces to the handover zone and a subsequent transport to workstations. The required parts are picked from the high-rack storage area (the storage spaces are illustrated as rectangles) and commissioned to the handover zone. From there they are transported to the workstations (illustrated as circles).

According to Kofler et al. (2010), the high-rack warehouse has been identified as a particular bottleneck in the material handling process and this effect is amplified by an increasing growth of production volume. In addition to throughput considerations maximizing the number of parts that can be commissioned during a day in the warehouse, it is also relevant to synchronize the picking with the upstream in-house transport and production processes. The inter-dependencies of warehousing with the other activities do not allow an isolated optimization of storage assignment and picking and requires a holistic approach (Vonolfen et al., 2012b).

3.3.2 Integrated Simulation and Optimization

The aim of this work is to investigate interrelations between storage assignment, picking and in-house transport. For that purpose, individual models dealing with these different aspects of the system are integrated as illustrated in Figure 3.8. An advantage of this integrative approach is, that previously developed simulation and optimization models can be re-used an integrated.

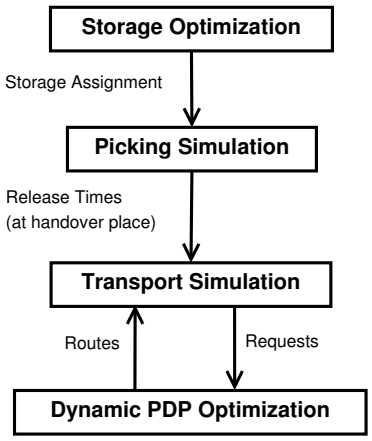

Figure 3.8: Information flow between the simulation and optimization models for warehousing and in-house transport (cf. Vonolfen et al. (2012b))

The storage optimization model creates a storage assignment on the basis of historical order data. This storage assignment is provided to a picking simulation model which is used to investigate the dynamic interactions between the pickers in the warehouse and to determine the exact time the commissioned materials arrive at the handover zone. These release time are provided to a transport simulation model which simulates the dynamic arrivals of these items. The in-house transport of the dynamically arriving items from the handover zone to the workstations is optimized using an online routing algorithm.

The main system objective is to minimize the total makespan which is the time until all items have been picked from the warehouse and delivered to the workstations. It is not only required to optimize the storage assignment, picking, and transport activities individually but to consider their inter-dependencies to reach an overall increased throughput from the warehouse until the items arrive at the production line. In the following, the individual system components will be examined. The storage optimization and picking simulation were investigated by Kofler et al. (2010) as well as Kofler et al. (2011) while the transport simulation and optimization were the main contribution in the context of this thesis.

Storage Optimization

Storage location assignment problems (SLAP) focus on slotting which is the assignment of storage spaces to products. When minimizing the effort of picking individual products, the travel time is an important factor and accounts for the largest part of the total effort.

Two measures for an evaluation of a storage assignment according to picking effort are pick frequency (PF) and party affinity (PA) of products were considered as proposed by Kofler et al. (2010). The evaluation is based on historical picking order data and the current storage assignment.

On that basis, the total PF score considers the number of times a product p_i was retrieved (represented by the function $orders(p_i)$) weighted by the average distance to the origin where the parts are commissioned. The distance function $dist(s_1, s_2)$ is calculated according to the warehouse layout for all storage locations. The average distance required to pick a product is calculated by considering all locations s in the set $L(p_i)$ where p_i is stored. The formula can be stated as following (Kofler et al., 2010):

$$PF = \sum_{i=0}^{n} orders(p_i) \cdot \frac{\sum_{s \in L(p_i)} dist(s, origin)}{|L(p_i)|}$$

Assuming that multiple products are usually picked together, the PA score considers the average distance between parts that were picked together. The number of times two products p_i and p_j have been picked together is represented by the $affinity$ function based on the historical order data. The affinity is weighted by the average distance of all storage locations s_k where p_i and s_l where p_j is stored (Kofler et al., 2010):

$$PA = \sum_{i=0}^{n} \sum_{j=0}^{n} affinity(p_i, p_j) \cdot \frac{\sum_{s_k \in L(p_i)} \sum_{s_l \in L(p_j)} dist(s_k, s_l)}{|L(p_i)| \cdot |L(p_j)|}$$

To evaluate the quality of a storage assignment based on historical picking order data, the total PF and PA scores are combined to a total weighted quality (Kofler et al., 2010):

$$quality = \alpha * PF + \beta * PA$$

The value of α and β determines the strength the turnover and affinity influence the storage assignment evaluation. This evaluation function has been investigated by Kofler et al. (2011) and the storage optimization algorithm is based on that. It is a simulated annealing (SA) algorithm where the

neighborhood is defined as swapping the contents of two storage locations. Appropriate parameter settings were determined as well as convergence and runtime behavior.

Picking Simulation

After the storage assignment has been optimized, the release times of the picked items are determined by a picking simulation model considering the dynamic interactions between the pickers in the warehouse. The simulation model was presented by Kofler et al. (2010) and was implemented in the simulation software AnyLogic©.

The model is based on the warehouse layout including the high rack storage area organized into aisles and the paths leading to the handover zone. In total, there is a pool of four human workers and two forklifts picking the items. The two bottom floors of the high-rack are served by the human workers while the forklifts serve all upper rows. The time required to pick an item from the high-rack storage is based on real-world observations.

The considered dynamic interactions between the pickers concern the blocking of aisles. Two forklifts cannot be in the same aisle at the same time. Additionally, a forklift cannot move past a human worker in an aisle. The blocking times influence the total picking times and thus the arrival time at the handover zone.

A fixed set of picking orders is simulated and the products of the orders are retrieved according to a certain picking schedule and commissioned at the handover zone. Two different picking schedules are investigated in terms how they influence the subsequent transport. In the clustered schedule, orders are grouped by their target manufacturing area leading to potential benefits for the subsequent in-house transport because pallets can be grouped by their target area. In the balanced schedule, orders are shuffled randomly reducing the chance for congestion in the aisles.

Transport Simulation and Optimization

After the items have been picked in the warehouse and commissioned at the handover zone, they are transported to the workstations. The in-house transport is modeled as a dynamic pickup and delivery problem.

There are two different types of transport requests. It is assumed that the items to be delivered to the workstations arrive dynamically at the handover zone after they have been commissioned. These arrival times have been determined in the picking simulation. Additionally, there are backhaul transports which are items that are delivered from the workstations back to

the warehouse. These requests are assumed to be known in advance and assumed to be distributed uniformly during the day.

The transport simulation optimization model is based on the generic pickup and delivery (PDP) model presented in Section 3.1.2. A standard PDP formulation is used with a distance matrix that was calculated based on the factory layout. Time windows are not considered and fixed service times are used. The transport requests are revealed dynamically according to the item arrival times. The objective is to sequence the transport requests in such a way, that the total travel time is reduced.

In terms of route optimization, an online genetic algorithm (GA) is applied. The applied GA uses mutation and crossover operators proposed by Potvin and Bengio (1996); i.e. the one-level exchange mutator, two-level exchange mutator, route-based crossover and sequence-based crossover. They are implemented using a route-based encoding which is examined and compared with other encodings by Vonolfen et al. (2011b). The initial generation is obtained by using a push forward insertion heuristic (Li and Lim, 2001).

An online optimization approach is applied which means that whenever a new transport request arrives (i.e., an item has been commissioned in the picking simulation and is ready for transport from the handover zone to the workstation), the population is updated and the new request is inserted into each individual.

In terms of algorithm parameters, a population size of 50 is used with a tournament selection and a mutation probability of 5%. Whenever a new request arrives it is inserted into each individual of the population using the best insertion heuristic and 50 generations are performed to compute a new route plan.

3.3.3 Conclusions

Based on the total lead time of the picking and in-house transport processes, two main influence factors have been investigated. The results presented in Section 6.2 indicate, that especially the picking schedule influences the downstream in-house transport process as well as the efficiency of the storage-assignment strategy in terms of congestion issues.

By considering only the warehousing process the results indicate, that using a balanced picking schedule together with combined PF and PA slotting would lead to the lowest average picking duration in the warehouse.

However the picture changes when also considering the down-stream in-house transport process. The average picking durations significantly increase in the warehouse when using a clustered compared to a balanced schedule. However, the average transport durations significantly decrease and the sav-

ings in transport lead to a better overall material process. This can be explained by the fact that clustering the picking orders by target workstations offers a higher potential for consolidation in the in-house transport. Especially in combination with PF/PA storage assignment, the likelihood for congestion increases.

A significant reduction in total makespan can be observed for all but the PF and PA strategies. When using solely turnover (PF) or affinity (PA) strategies, congestion becomes a major issue which is mitigated by combining the two factors to combined strategies or by using random slotting.

In general it can be concluded, that especially the picking schedule has a potentially large influence on the subsequent in-house transport activity. When changing the picking schedule, an appropriate storage assignment strategy has to be chosen in order to avoid congestion issues and consider turnover as well as affinity of parts.

Two interesting future extensions of the integrated model can be identified. Firstly, a production scheduling model could be integrated determining the time the items have to be delivered to the workstations. Currently, no time windows for the in-house transport have been considered. Secondly, the picking sequence could be optimized separately. So far only random and clustered picking sequences have been tested. In general, the combination and integration of several simulation and optimization models is a potentially fruitful research direction since the results indicate that an independent optimization of individual processes might lead to globally sub-optimal results.

3.4 Vendor-managed Inventory for the Distribution of Groceries

Vendor-managed inventory combines inventory management and routing and offers various degrees of freedom to the vendor which has access to the inventory information and makes decisions about the replenishment of goods. When considering large-scale stochastic and dynamic problem variants, efficient algorithmic strategies are required to deal with the problem complexity.

A simulation-based evaluation and optimization approach for modeling and optimizing stochastic and dynamic inventory routing problems with a large number of customers and products is presented. The methodology is validated with real-world scenarios generated from data provided by an Austrian retailer who delivers fast moving consumer goods to supermarkets from a central depot. Different scenarios are evaluated and a sensitivity analysis on different exogenous and endogenous impact factors is performed.

It is an example of applying simulation optimization as a scenario technique.

This section is based on a previously published study of Vonolfen et al. (2013a) where the methodology and results were first presented and from which several parts are paraphrased in the following. In the context of this thesis, the main contributions are the development of the two-stage simulation optimization approach as well as the generation of replenishment policies which will be examined in Section 4.2.

3.4.1 Context and Motivation

Vendor-managed inventory (VMI) is a well-known concept in supply chain optimization and has been applied successfully by various companies especially since the popularization by Wallmart in the 80ies (cf. Waller et al. (1999)). The mathematical formulation as an optimization model capturing this integration of inventory replenishment and routing is the inventory routing problem (IRP) which was first formulated by Bell et al. (1983).

Despite the various successful applications of this concept in practice, it remains an algorithmic challenge to solve high-dimensional inventory routing problems. The IRP can be seen as a generalization of vehicle routing problems. In addition to making routing decisions and minimizing distribution costs, the vendor also has to decide about replenishment while considering storage costs as well as service quality. This integration of routing and inventory replenishment leads to a high problem complexity and usually a longer planning horizon is considered.

When modeling and optimizing real-world problem formulations, usually several specific side-constraints have to be considered which led to a large number of different variants; a logistical overview is provided by Moin and Salhi (2007) and a survey about models and algorithms is provided by Bertazzi et al. (2007).

Especially the consideration of stochastic demand patterns is an active research topic while it could be stated that it is an important aspect of many real-world scenarios. The literature is still relatively scarce in terms of models and algorithms for stochastic IRPs. It remains an open research issue to consider large-scale instances with many customers and products.

Several heuristics are integrated in a methodological framework to model and optimize large-scale stochastic IRPs with a focus on the application in retailing. Specific to retailing is the fact, that a large number of fast moving consumer goods have to be considered while the availability of each single good influences the service quality and the product usage is uncertain. A simulation model is presented to account for the stochastic and dynamic aspects regarding product usage and is coupled with the optimization approach.

The methodology is applied to large-scale instances that are based on real-world data provided by an Austrian retailer that delivers fast moving consumer goods to supermarkets. The simulation optimization approach allows the investigation of several scenarios to perform a sensitivity analysis of endogenous and exogenous influence factors. A novel aspect in this context is the fact that mixed scenarios are considered where only part of the customers are transitioned to VMI while the remaining customers keep ordering themselves.

3.4.2 Problem Formulation

The model used in this work is based on the real-world case study in the area of retailing as an example of a high-dimensional IRP with stochastic product usages (cf. Vonolfen et al. (2013a)). In the classification scheme of Bertazzi et al. (2007), the model can be characterized as following:

- The inventory holding costs are not considered since the available storage in the supermarkets is limited to the sales area
- The product consumption rates are stochastic and follow given distributions which have been determined based on historical order data
- The product consumption takes place at discrete time steps $t = 0, 1, ...$ A time step in the simulation represents a day of operation. In each step, the daily product consumption is sampled from the probability distributions
- The product consumption varies over time. Concretely, weekly fluctuations have been considered based on the historical data

A novel aspect is, that the problem formulation allows mixed models that contain both VMI (N) as well as order-based (O) customers that are served by a homogeneous fleet of vehicles (V), each with a capacity C_v. The products P are distributed from a central depot and for each product $p \in P$ a storage capacity S_{np} is given for each VMI customer $n \in N$.

For the VMI-customers ($n \in N$), the product consumption rates are given separately for each weekday $w \in [1, 7]$ and product p resulting in a joint probability distribution U^w_{np} for the product consumption rates. The consumption rates do not change over time leading to a fixed weekly pattern. The vendor can measure the inventory level x^t_{np} and decides the amount d^t_{np} to replenish for each product and VMI customer every day. The number of out of stock situations s is the number of products over time where $x^t_{np} = 0$.

The order-based customers ($o \in O$) place a set of fixed orders F^t which contain the ordered quantities of the individual products d^t_{op}. These orders cannot be influenced by the vendor and have to be served. The orders are placed according to a classical threshold-based ordering strategy that considers a certain safety-stock.

The sets of the fixed orders (d^t_{op}) as well as the VMI replenishment (d^t_{np}) are combined into vehicle routes R^t whose length L_r can be calculated using a distance matrix based on the road network.

The objective function can be stated as following which aims at minimizing distribution costs while maintaining a certain service quality over a given planning horizon:

$$\begin{aligned} &\min \alpha * |V| + \beta * \sum L_r + \gamma * s \\ &s.t.\ x^t_{np} + d^t_{np} \leq S_{np}, \\ &\quad \sum_{d^t_{np} \in r} d^t_{np} + \sum_{d^t_{op} \in r} d^t_{op} \leq C_v, \\ &\quad s < s_{max} \end{aligned} \tag{3.2}$$

It consists of a weighted sum of the fleet size $|V|$, the total driven distance ($\sum L_r$) and the out of stock situations s. The weights were set as $\alpha = 3000$, $\beta = 2$, and $\gamma = 100$ after discussion with the domain experts. Feasible solutions fulfill the inventory capacity constraints regarding the storage at the VMI customers S_{np} as well as the vehicle capacity constraints C_v regarding the deliveries.The number of out of stock situations s cannot exceed a certain predefined limit s_{max} which determines the required service quality.

3.4.3 Simulation Optimization as Scenario Technique

In the context of this study, simulation optimization is applied for optimizing operative decisions as well as to support tactical decision making by evaluating several different scenarios. Therefore, the motivation to apply a simulation optimization approach to the stochastic inventory routing problem variant is twofold.

On the one hand, the proposed methodology allows the investigation of high-dimensional scenarios by coupling a simulation model with several heuristics. The IRP can be considered to be much more complex than the VRP. In addition to routing also resupply decisions have to be made and the planning horizon is longer. The optimization of long-term effects is complicated by the uncertainty coming from stochastic demand distributions.

Discrete-event simulation, especially in combination with metaheuristics, is a powerful methodology to model high-dimensional dynamic stochastic routing problems that would be hard to treat in an analytical way.

On the other hand, the motivation to apply simulation optimization stems from the fact that several scenarios can be investigated to evaluate the influence of exogenous and endogenous factors in the context of tactical decision making. The simulation environment allows the consideration of *what-if* scenarios and performing a sensitivity analysis to support the decision making process.

The simulation optimization model is based on the generic template for inventory routing problems presented in Section 3.1.2. A difference from the standard IRP formulation is the fact that mixed scenarios are possible including both customers that are served using VMI as well as customers that place orders themselves.

Regarding the optimization process, the decision making is split in two stages. In the first stage, the replenishment decisions are made. Based on the replenishment and also on the orders from the order-based customers, the tours are created. These decisions are made in each time step of the simulation which represent days of operation.

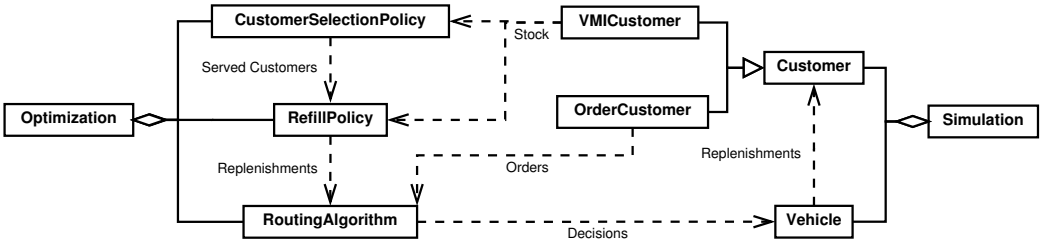

Figure 3.9: Schematic illustration of the information flow between the entities in the simulation optimization approach

Several heuristics are combined in the optimization approach as illustrated in Figure 3.9. The replenishment decisions are made using two policies. The first policy selects the customers to be served while the second policy determines the amounts of products to be refilled at the selected customers. The generation of these policies by means of simulation-based evolutionary policy search will be detailed in Section 4.2.

At each step of the simulation, after the replenishment has been determined, a standard CVRP problem instance is created based on the VMI replenishments as well as the placed orders. Concretely, the tours are calculated using a push forward insertion heuristic (Solomon, 1987). In principle, however, every routing algorithms capable of solving a CVRP can be applied.

The separation of these decisions potentially reduces the optimization potential since decisions made in the first stage determine the constraints for the routing. However, by dividing the decisions the problem complexity is reduced. To mitigate the loss of optimization potential, heuristic information about how the decisions influence the routing is added to the replenishment decision which will be detailed in the description of the replenishment policies in Section 4.2.

The two-stage decision process allows a separation of the problem to handle the complexity. While the evolution of the policies is computationally complex, they can be applied efficiently to make decisions once evolved. In combination with a routing heuristic, the proposed approach scales to large problem sizes with many customers and products.

3.4.4 Conclusions

Different endogenous and exogenous factors have been evaluated by applying a simulation-based scenario analysis using the presented methodology. Using a simulation optimization approach, a sensitivity analysis can be performed and different scenarios can be evaluated to aid in tactical decisions by evaluating optimization potentials. The conclusions are based on the detailed results which are listed in Section 6.3.

The most significant savings potential that was identified is the balancing of the resource utilization in cases of fluctuating demand patterns. This phenomena is illustrated in Figure 3.10. Without applying VMI, the fluctuation in the demand is transferred directly to the resource utilization. Even when only half of the customers are transitioned to a VMI, this fluctuation can be smoothed leading to a lower peak resource utilization by distributing the replenishment more equally. In the scenario with 100% VMI customers, the demand is nearly equally distributed over the week.

As an endogenous influence factor, the service quality was investigated. In the presented case-study, the service quality has a large influence factor on the distribution costs. When demanding 95% instead of 99% product availability, the fleet size can be reduced by 40.82%.

In terms of exogenous factors, the size of the product portfolio of a supermarket and as a result of the product demand had the greatest impact on the total costs. The results indicate, that switching supermarkets with the highest demand brings the largest benefit. Clustering customers geographically only showed limited benefits while creating a single large cluster was better than creating multiple smaller clusters. In the considered scenario, the largest optimization potential is the mitigation of the demand fluctuation explaining the large influence of the supermarket size.

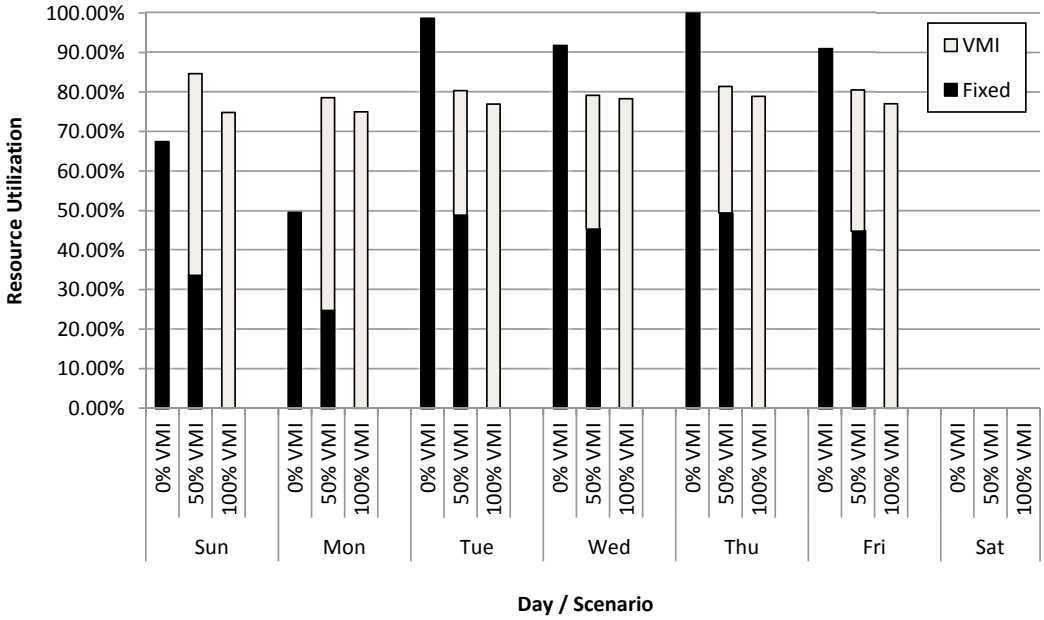

Figure 3.10: Resource utilization for the different scenarios. The fluctuation in resource utilization results from the weekly demand pattern and is smoothed in the VMI scenarios (cf. Vonolfen et al. (2013a)

Summarizing, the evaluation of different scenarios and the analysis of their optimization potential was performed by applying a simulation optimization approach. A sensitivity analysis of diverse influence factors can help understand the characteristics of a problem environment and aid in tactical decision processes. The optimization algorithm is used to evaluate the optimization potential of different scenarios in that case. For the considered case-study, the routing algorithm in combination with simulation-based generation of replenishment policies that are specialized to each scenario offers a heuristic approach do deal with high-dimensional scenarios based on realistic models.

The potential for future work is mainly in the practical application of the presented methodology. Different case-studies can be considered in the context of inventory routing. For instance, Vonolfen et al. (2011a) considered a fictional scenario for the collection of waste using electric vehicles based on the presented methodology. It would be interesting to consider increasingly dynamic inventory routing environments where simulation optimization is a feasible approach.

Chapter 4

Algorithmic Generation of Specialized Routing Policies

As outlined in Chapter 2, specialized solution methods are required depending on the problem properties of dynamic vehicle routing problems. For instance, Bertsimas and Van Ryzin (1991) pointed out that for light and heavy traffic situations completely different dispatching policies are required. Generally, depending on the degree of dynamism and the system objective, different solution methods are applied as Larsen et al. (2007) noted.

The aim of this chapter is to present an algorithmic framework to generate specialized policies that are adapted to the problem environment and can be combined and integrated with other solution methods. The main approach followed is an evolutionary policy search based on a black-box simulation model and an adaptable heuristic policy.

Apart from the case-studies of Beham et al. (2009b) and van Lon et al. (2012) illustrating the potential, the generation of heuristic policies has not received wide attention in the context of dynamic vehicle routing problems so far. The aim of this work is to explore the generation of several heuristic policies for different variants of dynamic vehicle routing problems with practical applications.

Based on the algorithmic framework outlined in Section 4.1, three case-studies are presented: the generation of replenishment rules for inventory routing (Section 4.2), dispatching rules for dial-a-ride (Section 4.3), and waiting strategies for pickup and delivery problems (Section 4.4).

4.1 Simulation-Based Evolutionary Policy Search

When solving dynamic vehicle routing problems, the general aim is to find a *routing policy* specifying what action should be taken as a function of the state of the system that evolves over time (cf. Psaraftis (1995)). In the past, mostly human-designed heuristic policies have been applied to dynamic vehicle routing problems focusing mainly on average-case performance. This process usually requires extensive algorithm design and testing as well as domain expertise.

Within the vision to create heuristic methods that are self-adaptive in terms of problem characteristics, an automation of generating and adapting heuristic policies is needed. Furthermore, heuristics that are specialized to certain problem characteristics have the potential to perform better than human-created heuristics which usually are designed to perform well on a wide range of problem instances (cf. Burke et al. (2013)). For instance, using HeuristicLab, the automatic generation of heuristics has been successfully applied in practical contexts for the evolution of priority policies for production fine-planning (Pitzer et al., 2011), control policies for power flow in smart grids (Hutterer et al., 2013) as well as dispatching policies for dial-a-ride problems (Beham et al., 2009b). However, within the field of dynamic vehicle routing, automated algorithm design has received only limited attention so far.

The presented methodology aims at a semi-automated generation of heuristic routing policies based on evolutionary policy search and a simulation model. The search space as well as the components for the generation and adaption of the routing policies is defined by a human expert. As Burke et al. (2013) points out, previous studies have shown that human experts are able to provide good building blocks that can be assembled and adapted in superior ways by algorithmic strategies.

The goal is to generate re-usable heuristics that can be successfully applied online to previously unseen problem instances. Since in dynamic environments the computation time is usually limited, it makes sense to shift the generation to an offline training phase. The heuristic policies are adapted and generated on a set of representative training instances which have similar characteristics as the targeted online environment by means of a reinforcement learning mechanism.

The generated heuristics are applied within a pre-defined algorithmic framework and tackle a part of the overall problem that needs to be adaptive in terms of problem characteristics. For instance, specialized waiting policies

can be applied within a generic dynamic routing framework. As Burke et al. (2009) note, this combination of human domain knowledge and algorithmic policy generation has proven fruitful and the automatically designed policies complement human designed heuristics.

Intentions to automate the generation and adaption of heuristics are not new and have been investigated from different perspectives in the past. As Pappa et al. (2013) point out, it can be generally observed, that research efforts in different fields including operations research, optimization, and machine learning ultimately evolved from algorithm / parameter selection and control to the automated generation of algorithms.

First approaches to automatically adapt and generate heuristics can be dated back to seminal work of Fisher and Thompson (1963) who combined several dispatching rules for production scheduling in a probabilistic learning phase by a biased selection. Several studies to generate dispatching rules such as the work of Hart and Ross (1998) followed in the 90ies.

Within research on evolutionary algorithms, especially parameter setting and control strategies were investigated including self-adaption (Bäck and Schwefel, 1993) and meta-optimization (Grefenstette, 1986). This lead to the generation of full evolutionary algorithms (Oltean, 2005).

A rather recent stream of research, which is surveyed by Burke et al. (2013), are hyper-heuristics which are based on high-level heuristics who operate in the search space of heuristics based on low-level heuristics or heuristic components. Initially hyper-heuristics were mainly concerned with the selection of low-level heuristics but the definition was later extended to the generation of new heuristics based on algorithm components. Within this framework, the presented methodology can be categorized as a generative hyper-heuristic since meta-heuristics are applied to search in the space of routing policies within an offline learning phase.

Successful approaches for the automated generation and adaption of heuristics usually combine several elements from machine learning, artificial intelligence, and operations research.

4.1.1 Methodological Foundations

The presented methodology is based on direct policy search using evolutionary computation and reinforcement learning. The generated heuristic policies are integrated in heuristic frameworks typically combining several approximations to tackle the inherent problem complexity of dynamic vehicle routing problems. Policy approximations, reinforcement learning, and evolutionary computation are the theoretical foundations which are combined to a methodology for the semi-automated generation of heuristic policies.

Policy Approximations

Due to the complexity of dynamic vehicle routing problems, approximate policies are used for solving practical variants. A formal notation for defining policies as a decision function of the system state is provided by the unified modeling framework for dynamic decision problems proposed by Powell et al. (2012) which is based on the following elements:

- The *state variable* captures the state of the system at each time step t. It includes the resource state R_t which can be referred to as the physical system state (e.g., vehicles), the information state I_t which includes all other information required to make decisions (e.g., prices), and the knowledge state K_t including beliefs about random variables (e.g., forecasts):

$$S_t = (R_t, I_t, K_t) \tag{4.1}$$

- The state evolves over time which is described by the *state transition function*. It takes into account the current state S_t, the decision variable x_t and the newly arriving exogenous information W_{t+1}:

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}) \tag{4.2}$$

- The decisions are made according to a *decision function* X^π based on a policy π, that maps the current state to decisions:

$$x_t = X^\pi(S_t) \tag{4.3}$$

- The total expected costs are minimized over all time steps that incur by making the decisions x_t at state S_t. The aim is to find an optimal policy π:

$$\min_{\pi \in \Pi} \mathbb{E} \sum_{t=0}^{T} C(S_t, x_t) \tag{4.4}$$

When solving large-scale dynamic decision problems, Powell (2007) points out that there are three curses of dimensionality. This refers to a rapid, often exponential, growth of problem complexity with increasing size which affects the state variables (S_t), the random variables (W_t) and the decision variables (x_t). As a result, real-world problems become complex in terms of memory, modeling and runtime requirements.

Dynamic vehicle routing problems fall into the category of problems that rapidly grow in complexity requiring approximate methods such as heuristics. As a result, approximations are applied to cope with the resulting complexity. In fact, this offers a new viewpoint on the solution methods presented in Chapter 2 where most of them apply one or more of the following approximations.

Powell et al. (2012) identified four fundamental types of policy approximations and characterizes solution methods for sequential decision problems in general and dynamic vehicle routing in particular accordingly:

- Myopic cost function approximations consider the decisions x_t to be taken given the current state S_t and minimize the immediate costs $C(S_t, x_t)$ while not regarding long-term effects caused by these actions.

- Lookahead policies optimize a finite horizon into the future with the aim of considering long-term effects in a rolling-horizon manner. Future events can be either approximated deterministically or accounted for explicitly by means of a stochastic model.

- Value function approximations are based on estimations of the value of being at a current state S_t which consists of the immediate costs $C_t(S_t, x_t)$ and the expected future costs when following a certain policy. This avoids evaluating all possible future states.

- Policy function approximations are based on analytic functions mapping actions x_t to a given state S_t directly without solving an underlying optimization problem. This is especially suitable in cases where $X^\pi(S_t)$ has a structure that can be derived for example from domain knowledge.

Usually multiple approximations are combined in solution methods tackling real-world problem formulations as Powell et al. (2012) notes. For instance, a policy function approximation can be used within a lookahead policy. Such an example would be a waiting policy incorporated in a rolling-horizon reoptimization approach as proposed by Ichoua et al. (2006).

Reinforcement Learning

Apart from the algorithmic challenges that arise from the complexity of real-world applications, the modeling of large dynamic and stochastic systems is far from trivial. This is known as the curse of modeling (cf. Gosavi (2003)). A closed-form model often does not exist or would be too difficult to obtain due to the complexity or incomplete knowledge for many practical applications. As a result, rich dynamic vehicle routing problems are usually studied by means of simulation models.

In situations where no closed-form is available but trial and error interactions with a simulation model can be made, reinforcement learning is a feasible method to solve sequential decision tasks in general because it does not require an analytical model (cf. Grefenstette et al. (1990)). The basic principle is, that a reward is received for actions taken in the environment. The rewards can be given after each action or sparsely after several actions have been performed. The policy is adapted according to the feedback.

An advantage of reinforcement learning is that no analytical (theoretical) model is required. Instead, the governing variables are simulated which is a feasible approach if no tractable model of the system is available since sufficient domain knowledge would be too costly to obtain due to the high complexity or is not accessible (Gosavi, 2003). The system state is implicitly represented in the simulation without requiring an analytical model.

For a deep introduction investigating the links between reinforcement learning and dynamic programming, the reader is referred to Busoniu et al. (2010). In particular, reinforcement learning techniques are applied in the field of approximate dynamic programming to solve large-scale sequential decision problems by learning approximate value functions; Powell (2012) outlines the links as well as the historical development.

Well-known classical approaches in the area of reinforcement learning include algorithms such as Q-learning (Watkins and Dayan, 1992) that rely on the expected future reward (i.e. the value) of actions performed in certain states (state-action pairs) and iteratively build up an action-selection policy by exploration of the state-action space.

An alternative approach is direct policy search where optimization techniques are applied to find a policy maximizing the expected reward. While Q-Learning works in the state-action space, direct policy search works in the policy space. An advantage is, that compact representations can be used for policy functions and their structure can be derived from domain-knowledge (Whiteson, 2012). Alternatively, the structure of the policy itself can be optimized based on a fixed grammar.

In terms of computational complexity, several empirical studies confirmed

that direct policy search successfully scales to large state spaces as Whiteson (2012) point out. Important success factors are rooted in the policy representation. A direct policy representation allows a higher level of abstraction by grouping several states. In combination with a selective representation, that only stores successful policies, the complexity of the state space can be reduced significantly (Moriarty et al., 1999).

However, efficient and robust search techniques are required for policy spaces which can get potentially very large. Applying evolutionary computation to reinforcement learning has proven fruitful for solving sequential decision tasks, as Moriarty et al. (1999) and recently Whiteson (2012) pointed out.

Evolutionary Computation

Evolutionary algorithms are a class of metaheuristics that applies the principle of natural evolution to optimization, or more generally to study adaptive systems. A population of candidate solutions (individuals) is evolved iteratively by generating offspring from selected parent solutions. The main driving factors is evolutionary selection which steers the process towards a defined system objective and follows the Darwinian principle of "survival of the fittest". There are several variants of evolutionary algorithms that focus on different application areas. For a detailed overview on evolutionary computation the reader is referred to Eiben and Smith (2010) as well as Affenzeller et al. (2009).

Being a generally applicable search technique, evolutionary algorithms consist of problem independent and problem specific parts. The problem specific parts are based on a representation of the individuals which is called genotype. The genotype defines the search space in which the evolutionary process takes place (cf. Eiben and Smith (2010)). For instance, there exist multiple possible representations for vehicle routing problems and selected ones been examined by Vonolfen et al. (2011b).

The main steps of an evolutionary algorithm are illustrated in Figure 4.1. Problem dependent steps are the evaluation, crossover and mutation which have to be defined according to the representation. All other steps are problem independent and only rely on the evaluation (fitness) value of an individual.

Evolutionary algorithms are a population-based methods and evolve a set of solutions (population of individuals) simultaneously. The first step is an initialization of the population either randomly or by means of a construction heuristic. For the vehicle routing problem several construction heuristics exist such as the savings heuristic (Clarke and Wright, 1964) or the push-

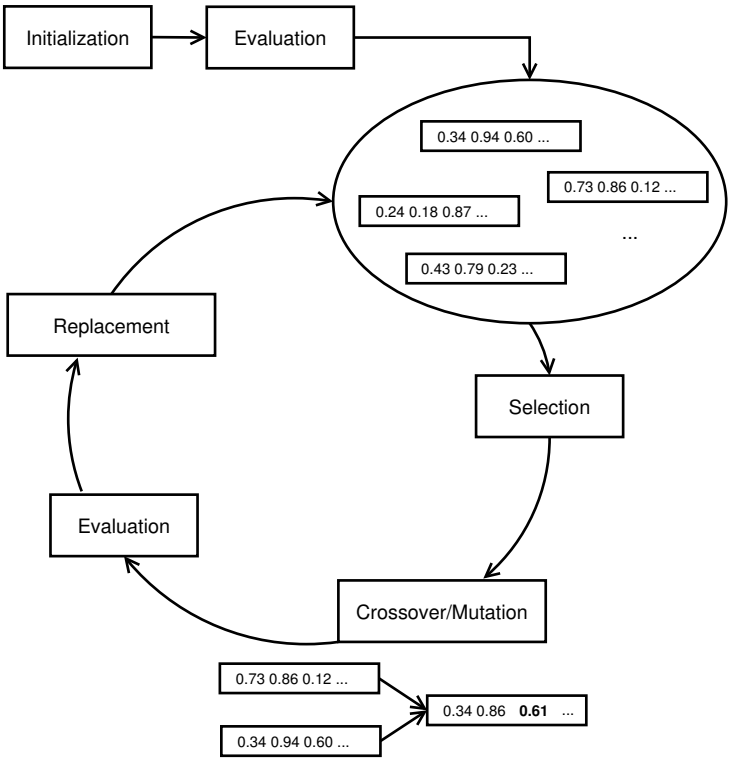

Figure 4.1: Basic scheme of an evolutionary algorithm

forward insertion heuristic (Solomon, 1987). Each individual is evaluated and assigned a fitness.

After the initialization phase, an iterative process is performed that mimics the principles of natural evolution. Each iteration, parent individuals are selected to produce offspring individuals. The offspring is produced by using problem-specific crossover and mutation operators. Crossover operators combine the genetic information of two or more individuals while mutation makes small, undirected changes. The newly created offspring individuals replace individuals in the parent population. Then, another iteration is performed by generating new offspring individuals.

The selection is solely based on the fitness value and is the main driving factor in the evolutionary process. There are several different selection methods. When using proportional selection, individuals are selected with a probability proportional to their fitness. In rank selection, the individuals are ordered according to their fitness and assigned a fitness value according to their rank. For instance, linear and non-linear fitness assignment can be used. In tournament selection, a fixed number of individuals is selected and the fittest among them is chosen. In general, the applied selection mechanism as well as its parametrization has a strong impact on the selection pressure and as a result on the dynamics of the evolutionary process (cf. Affenzeller et al. (2009)).

From the selected parent individuals, a set of offspring individuals is created by crossover and mutation which are specific to the representation of the individuals. Crossover aims at combining genetic information of two or more individuals while mutation performs small undirected variations.

After evaluating each offspring solution, the newly generated offspring replaces individuals of the current population. Different replacement schemes can be used. Eiben and Smith (2010) distinguish between age-based and fitness-based replacement. An example of an age-based stagey would be generational replacement where the whole parent population is replaced by the offspring. Fitness-based replacement schemes consider both the parent and child offspring and in that case the same selection operations as for parent selection can be applied. An extension that can be used with all replacement schemes is elitism which preserves a specified number of best individuals that are excluded from replacement.

This basic scheme of evolutionary algorithms is followed by several variants that focus on different application areas or apply certain algorithmic extensions. The most prominent variants are genetic algorithms, genetic programming, and evolution strategies. The main differences and application areas are outlined by Eiben and Smith (2010) which will be put in context of the presented methodology.

Genetic algorithms (GA) build on the efficient interplay of crossover and mutation and were introduced by Holland (1975) to study adaptive systems. Mutation introduces new genetic information which is propagated by crossover. The main theory behind genetic algorithms is the building block hypothesis which is defined for a canonical GA with binary representation. It proves, that short sequences with above average fitness (building blocks) are assembled to longer sequences in the evolutionary process. This theorem fails to hold for non-idealized GA variants, however should be considered when designing a GA for a particular application area. Naturally, these considerations rather have combinatorial than real-valued optimization in mind.

Genetic programming (GP) can be seen as a special variant or extension of a GA that aims at evolving computer programs which are usually represented as trees. There are some algorithmic differences compared to standard GAs such as independent execution of crossover and mutation and the use of large population sizes. GP has been successfully applied in various areas such as machine learning and symbolic regression and has proven to be a powerful automated problem-solving method. As a theoretical foundation, the GA schema theorem of Holland (1975) has been transferred to GP by Koza (1992).

Evolution Strategies (ES) are a different stream of EAs and have been developed independently from GAs for a long time (cf. Affenzeller et al. (2009)). ES have been designed explicitly for continuous optimization with a strong emphasis on mutation. As Eiben and Smith (2010) notes, a core concept is self-adaptivity introduced by Schwefel (1977) which means that the mutation step size is adapted during an optimization run. Focusing on real-valued representations, a main application area of ES is continuous parameter optimization.

4.1.2 Methodology

The combination of reinforcement learning and evolutionary computation allows the adaption and generation of heuristic policies by trial and error interactions with a simulation model. The resulting heuristics are policy function approximations which are embedded within a heuristic framework which usually combines several approximations to tackle the problem complexity.

The presented methodology is based on three main assumptions. Firstly, approximations are necessary to cope with the complexity of real-world sequential decision problems. Policy function approximations are suitable if their structure can be derived from domain knowledge. Secondly, real-world dynamic vehicle routing problems are mainly investigated by means

of discrete-event simulators since a closed-form analytic model is often not available. Thirdly, solution methods that are specialized to the characteristics of the problem environment are required for dynamic vehicle routing. The main hypothesis is, that evolutionary policy search based on a simulation model can overcome the curses of dimensionality when solving real-world dynamic vehicle routing problems by generating specialized policies.

The policies are evolved offline in a training phase and are applied in an online setting a-posteriori which has similar characteristics as the training instances. The assumption that the presented methodology can scale to practically relevant dynamic vehicle routing problems stems from the combination of simulation optimization and evolutionary reinforcement learning. The main success factors are overcoming the complexity of modeling dynamic vehicle routing problems as well as their computational complexity.

One measure to mitigate the complexity is a generalized policy representation which is based on a set of domain-specific features which can be viewed as functions of the current state grouping together several states. The features are derived from domain knowledge or alternatively, could be obtained by applying automatic feature generation (Fawcett and Utgoff, 1992). Using abstract features as input variables for the policies aims at mitigating the complexity of the considered problems as well as at evolving generalized policies by abstracting several similar state and thus reducing the state space complexity.

The adaption to the problem environment is performed by searching in the space of possible policy functions. Direct policy search allows a compact representation of the policies whose structure is either fixed and derived from domain knowledge or is flexible and subject to optimization itself. The search space thus can be defined either as a set of parameters for a policy with a fixed structure or as a set of formulas with a fixed grammar for flexible policies. Evolutionary algorithms are applied for searching in the space of policies, since they are global search techniques that are known to work well in large and multimodal search spaces (Affenzeller et al., 2009).

The found policies are combined with other approximate solution methods and tackle a certain aspect of the problem environment that needs to be flexible in terms of its characteristics. For instance, a specialized waiting policy can be combined with a general rolling-horizon routing algorithm. The waiting policy can be adapted in an offline training phase to different temporal or spatial properties. For clustered problem instances a different waiting policy is applied than for randomly distributed requests.

As Moriarty et al. (1999) note, there are important design considerations for the application of evolutionary policy search concerning the policy representation, the learning phase and the search algorithm.

Policy Representation

The policy representation defines a basic data structure for mapping states to actions. The representation of the policy strongly influences the learning phase as well as the search algorithm. Moriarty et al. (1999) outlines several options for representing policies in the context of evolutionary policy search. A main aspect for choosing the representation is the level of generalization.

Policies mapping states directly to actions range from table-based to rule-based and neural-network-based policies. Table-based policies explicitly map states to actions requiring an action to be specified for each state. In contrast, generalized mappings group together similar states avoiding an explicit consideration of each state. Rule-based policies contain condition-action pairs potentially allowing the creation of generalized rules. Neural network representations implicitly model the mapping function by interconnected neurons. In that context, especially the field of neuro-evolution has received considerable attention in the literature and is surveyed by Whiteson (2012).

In the proposed methodology, the policy functions are defined on a predefined set of n domain-specific features which are functions of the current system state ($f_i(S_t) \in F, i = 1..n$) and represent it in a higher level of abstraction. The features are synthesized into complex policies representing a mapping from features to actions. The evolved policy function approximation $A^\pi(F)$ returns an action x_t for the feature set F.

The policy representation, which defines the search space for direct policy search, can be either a fixed structure or a flexible structure. In terms of fixed structure, for instance a linear combination of the feature values can be considered. In that case, a vector with a fixed size is used where each element represents a parameter of the policy. On the other hand, flexible policies are based on a grammar without assuming a certain structure beforehand. The features are combined by algebraic and logical terms allowing the evolution of more complex policies. The relevant features are automatically selected and nonlinear relationships can be identified.

Learning

The learning phase deals with searching for parameters in the case of policies with a fixed structure and formula trees in the case of a flexible structure based on the feedback on the performance in a simulated dynamic vehicle routing environment. The learning is performed offline in a training phase and credit is assigned to policies for their resulting decision sequences. The aim is to find policies that are adapted to the characteristics of the problem environment and work well in similar, but previously unseen situations.

Reinforcement learning can be either performed offline or online. Online learning means to learn the policies directly in an operational environment while offline learning is performed in a simulation environment beforehand. As noted by Moriarty et al. (1999), evolutionary reinforcement learning requires the evaluation of hundreds or thousands of policies. Since experiments in a real dynamic vehicle routing environment are costly, it makes sense to perform offline training using a simulation model.

The offline learning process is performed on a set of training problem instances. Each instance represents a planning horizon which could be for example a day of operation. To draw conclusions about the performance of the evolved policies on similar but previously unseen instances, a set of test instances is used that are not used during the training phase. The set of instances can be obtained either from collections of standard benchmark instances or derived from historical data in practical case studies. The advantage is, that usually a large number of different instances can be obtained or generated for the training phase to cover as many as possible situations which is important in terms of robustness of the policies.

Robustness considerations are also an important aspect when it comes to credit assignment. Evolutionary policy search usually assigns the credit to a whole sequence of actions and implicitly distributes it among the individual actions by the evoloutionary process (cf. Moriarty et al. (1999)). In a dynamic vehicle routing environment the actions taken in a given time step strongly influence future decisions. As a result it is a difficult task to evaluate the value of each individual action and the feedback is given for each problem instance as a whole, implicitly assigning credit to the whole action sequence.

Resulting from offline training and implicit credit assignment, there are challenges related to evolutionary policy search that need to be considered. Moriarty et al. (1999) mentions the lack of explicitly considering knowledge about bad decisions and the tendency of decisions that had a low impact in the training phase to drift to random values. These issues mainly stem from the selective representation of the policies in the evolutionary search process. Only successful policies are kept and no explicit information about bad decisions is stored.

Another important consideration in the learning phase is the danger of over-adaption in the training-phase (see for instance (Hawkins, 2004)) which means that the evolved policies are over-adapted to the set of training instances and do not perform well on previously unseen instances.

Considering these aspects, it is important to cover a wide number of situations and provide a sufficiently large and diverse set of training instances as well as to explicitly mitigate these problems in the search algorithms to generate robust policies that can be successfully applied in an online setting.

Search Algorithm

The policies are generated by searching in the space of possible configurations based on the underlying representation in an offline reinforcement learning phase. Evolutionary algorithms (EA) have an empirical record of being a suitable algorithm for direct policy search (cf. Whiteson (2012)). The specific aspects of evolutionary algorithms regarding direct policy search will be outlined in the following including the evaluation of individuals, the used representation, and the applied variants.

The evaluation of the individuals is performed by simulating a set of training instances based on the simulation and optimization environment presented in Section 3.1. This principle is illustrated in Figure 4.2. Each individual represents a candidate policy either represented as a real-valued vector or as a formula tree. As detailed in Section 4.1.2, the evolved policy represents the flexible part of the solution method and is combined with other approximations. For instance, the illustrated real-valued vector could represent a parametrization of a waiting strategy. In that case, the goal would be to evolve a strategy that is adapted to the problem characteristics and yields to beneficial decisions in terms of scheduling the vehicle waiting times within the simulation optimization environment.

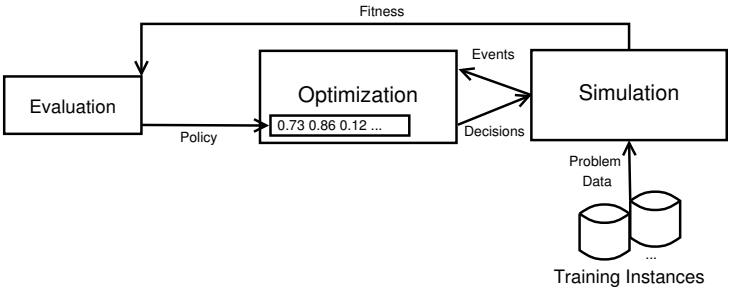

Figure 4.2: Simulation-based policy evaluation

In the context of direct policy search, two different representations will be considered. For policies with a fixed structure a real-valued vector, while for policies with a flexible structure a tree representation will be used. For these representations, well known genetic operations exist which can be used as a basis for the evolutionary policy search algorithm.

For real-valued vectors both discrete and arithmetic recombination is used (cf. Eiben and Smith (2010)). Discrete recombination merges two real-valued vectors by selecting a parent to inherit each position of the vector. On the contrary, arithmetic recombination averages the values of the parents. Mutation operations on real-valued vectors include one-position uniform mutation

where a single position of the vector is replaced with a random value that is sampled from a uniform position and all positions nonuniform mutation where each position of the vector is manipulated with a small Gaussian value.

For tree representations specialized crossover operations exist that merge the structure and values of several parent trees into an offspring tree (cf. Affenzeller et al. (2009)). A commonly used variant is sub-tree crossover which exchanges randomly chosen sub-trees of two parents creating two new offspring trees. Mutation operators for tree representations include replacing a randomly chosen node with a generated sub-tree or deleting sub-trees.

Based on these operations, two variants of evolutionary algorithms are applied to direct policy search based on their success factors. Evolution strategies are applied to the evolution of policies represented as real-valued vectors while genetic programming is applied to evolve policies represented as formula trees.

ES are known to work well in continuous parameter spaces. A main success factor is the adaptive mutation step size which steers diversification and intensification adaptively. On the contrary, simple greedy methods would quickly get stuck in a local optimal. As pointed out by Eiben and Smith (2010), empirical evidence shows that mutation step sizes should be larger in the beginning to allow exploration and then get smaller to intensify the search in promising regions. The adaption of the step size generally leads to a faster convergence. These investigations have been proven theoretically for certain idealized conditions as summarized by Bäck et al. (1991) and shown empirically for more complex problems. With continuous parameter optimization being a main application area, these aspects make ES a suitable strategy for searching parameters for policies represented as real-valued vectors.

GP has a large empirical success story in evolving computer programs that generate human-competitive solutions for various types of problems. The theory behind the algorithmic principles was developed mainly building on the schema theorem initially defined for GAs by Holland (1975). The schema theorem for GP proposed by Koza (1992) defines so called S-expressions as the basis for schemas which are basically a set of sub-trees. Applied to evolutionary policy search, the GP process builds larger and more complex policies from a set of building blocks which are sub-trees with a high fitness value. This allows the identification of non-linear relationships between the individual features and build up complex policies from these building blocks.

As stated in Section 4.1.2, algorithmic measures are taken to prevent an over-adaption to the training instances while not performing well on previously unseen data. The aim is to generate robust policies that work well for similar, but previously unseen problem instances.

A commonly investigated danger in GP is the phenomenon of code bloat. In the context of evolutionary policy search, it means that too complex policies would be evolved while simpler policies would also fulfill the task. This bears the danger, that the policies are not intelligent abstractions of decision making in the observed situations but are potentially over-adapted to the training instances. To mitigate this issue, the tree size is limited to avoid code bloat.

Another countermeasure to over-adaption that is applied both for ES and GP is a dynamic separation of the training set. In each iteration, only a part of the training set is used to evaluate the current individuals. However, the whole training set is used to select the best found policy.

The described methodology will be applied to evolve policies that are adapted to certain problem characteristics. These policies are incorporated in approximate solution methods and applied to practically relevant variants of dynamic vehicle routing problems.

4.2 Replenishment Policies for Inventory Routing Problems

Dynamic and stochastic inventory routing problems are complex sequential decision processes since inventory replenishment as well as routing decisions have to be made which are dependent on each other. This combination of routing and replenishment decisions while considering long-term effects makes the problem highly intractable requiring heuristic solution approaches. The scenario presented in Section 3.4 represents a high-dimensional inventory routing problem with stochastic product usage and requires the combination of several approximations to be tractable in terms of modeling effort and computational complexity.

Based on the simulation optimization approach presented in Section 3.4, a heuristic solution method applying several approximations is presented. Core of the method are approximate inventory replenishment policies that are generated using evolutionary policy search. This section is based on a previously published study of Vonolfen et al. (2013a). The main contribution in the scope of this thesis is the application of evolutionary policy search to inventory routing scenarios based on real-world data in the context of retailing.

4.2.1 Two-Stage Decision Process

The inventory routing problem combines inventory replenishment as well as routing decisions. As already discussed in Section 3.4.3, a two-stage algorithm is applied. In the first stage, replenishment decisions are made which means the amount of deliveries is determined for each product. In the second stage, the deliveries are sequenced to routes which are executed by a fleet of vehicles.

The presented heuristic approach combines several approximations in a rolling horizon manner. The planning horizon is one day of operation while future product consumption is accounted for by deterministic forecasts. At each time step of the simulation, which represents a day of operation, an approximate inventory and routing policy is executed which makes replenishment and routing decisions for the current day.

The policy has a fixed structure which has been derived from domain knowledge. It consists of several parameters which are set according to different scenarios by means of reinforcement learning. Core of the approach are two approximate policies for customer selection and product replenishment which are combined with a routing algorithm. The definitions are based on the problem model presented in Section 3.4.2.

The top-level policy combines the different elements and is illustrated in pseudocode in Algorithm 1. Input parameters are the set of VMI customers N, the vehicle capacity C_v, and a set of fixed orders F^t for non-VMI customers. The other two parameters ($CapacityUtlization$ and $Priority - Threshold$) are set according to scenario properties in the reinforcement learning process.

The policy basically works in a two-stage process which means that firstly replenishment and secondly routing decisions are made. However, to consider the effect on the routing in the first stage, the deliveries are inserted in a preliminary set of routes R immediately by using an insertion heuristic. As an insertion heuristic, the best possible insertion position in the preliminary routes is chosen. The preliminary set of routes R is initialized by applying a routing heuristic to the fixed set of orders F^t. This links the replenishment and routing decisions together within the two-stage process.

In the first stage, a fixed capacity c is determined that is available for replenishment which is a fraction of the total available vehicle capacity and is determined by the $CapacityUtilization$ parameter. The aim is a constant resource utilization during all days. Setting this threshold to a lower value implies the use of less capacity balancing the resource utilization since the parameter is a constant over the whole planning horizon.

While there is still capacity available, the customers to be replenished are

Algorithm 1 Approximate inventory repelnishment and routing policy (cf. Vonolfen et al. (2013a))

Require: N, C_v, F^t, *CapacityUtilization*, *PriorityThreshold*

$D \leftarrow F^t$
$R \leftarrow RoutingAlgorithm(D)$
$c \leftarrow (\sum C_v) * CapacityUtilization - \sum_{d_p \in F^t} d_p$
while $N \neq \emptyset$ AND $c > 0$ **do**
  select $n \in N$ according to **CustomerSelectionPolicy**
  $N \leftarrow N \setminus n$
  if $\lambda_n >= PriorityThreshold$ **then**
    select deliveries D_n for n according to **RefillPolicy**
    $D \leftarrow D \cup D_n$
    $c \leftarrow c - \sum_{d_p \in D_n} d_p$
    $R \leftarrow InsertionHeuristic(D_n, R)$
  end if
end while
$R \leftarrow RoutingAlgorithm(D)$

chosen according to a customer selection policy which chooses the customer with the highest priority from a set of customers. The priority calculation is a weighted average over a set of feature values. Each feature f_{ni} given for a customer n is weighted with a fixed factor a_i which is parametrized in the reinforcement learning process. Since the parameters a_i as well as the features f_{ni} are normalized in the interval $[-1, 1]$, the resulting priority value also lies in that interval. The following features are considered (Vonolfen et al., 2013a):

f_{n1} - Predicted number of days until the first product will run out of stock for customer n

f_{n2} - Predicted average number of days for all products to run out of stock in the inventory of customer n

f_{n3} - Number of days since the last delivery to customer n

f_{n4} - Total inventory capacity of customer n over all products ($\sum_p S_{np}$)

f_{n5} - The minimum required detour to integrate a delivery to customer n in the set of preliminary routes R

f_{n6} - The geographical isolation of customer n which is defined as the average distance to all other customers

While features f_{n1} to f_{n4} mainly concern the service quality, features f_{n5} and f_{n6} take into account the subsequent routing process. The priority λ_n for a given customer n is calculated according to the formula illustrated in Equation 4.5.

$$\lambda_n = (\sum_{i=1}^{m} f_{ni} * a_i)/m \tag{4.5}$$

If the priority λ_n lies above a certain limit specified by the $Priority - Threshold$ parameter, the products to be replenished at the chosen customer are determined by a refill policy which is listed in Algorithm 2.

Algorithm 2 Refill policy for a given customer (cf. Vonolfen et al. (2013a))

Require: c, n, t, *RefillThreshold*, *RefillBarrier*, *RefillFactor*

$b \leftarrow c$
while $P \neq \emptyset$ AND $b > 0$ **do**
  select $p \in P$ where min $OOSP(p, n, t)$
  $P \leftarrow P \setminus p$
  if $OOSP(p, n, t) < RefillThreshold$ OR $x_{np}^t < RefillBarrier$ **then**
    $d_p \leftarrow (S_{np} - x_{np}^t) * RefillFactor$
    $b \leftarrow b - d_p$
  end if
end while

The refill policy sorts the products p of the selected customer n by their estimated out of stock prediction (OOSP) based on the stock available at the current day t. While there is still capacity left, all products where the OOSP is below a given $RefillThreshold$ or the current item stock x_{np}^t is below a given $RefillBarrier$ are refilled to a certain level determined by the parameter $RefillFactor$. These three parameters are set in the reinforcement learning phase.

After the replenishment decisions have been made, the deliveries are combined in a set of roues by a routing algorithm. The preliminary roues are discarded and a full re-optimization of these routes is performed to utilize additional routing potential. As a routing algorithm, the push-forward insertion heuristic is applied. In principle, every routing algorithm could be used as outlined in Section 3.4.3.

4.2.2 Simulation-Based Parametrization

As described in the previous section, there are 11 parameters of the policy that are set according to the scenario characteristics: *CapacityUtilization*,

$PriorityThreshold$, a_i for $1 \leq i \leq 6$, $RefillThreshold$, $RefillBarrier$, and $RefillFactor$. Direct policy search based on reinforcement learning and evolutionary computation is applied to generate specialized policies for different scenarios. The policy has a fixed structure and is represented as a vector of parameters with length 11. Each element of the vector corresponds to a parameter of the policy and has a value range in the interval $[-1, 1]$.

For the optimization of the real-valued parameter vector a $\mu+\sigma$ evolution strategy was applied with $\mu = 1$ and $\sigma = 3$. A self-adaptive Gaussian mutation was used with learning parameters $\tau = 0.4$ and $\tau_0 = 0.4$. The process was repeated for 100 generations.

The greedy settings of the algorithm stem from the relatively long computation time of the simulation. Each individual has to be evaluated using several simulation runs due to the stochastic product usage. Each candidate parameter vector is evaluated on 3 test scenarios which have been specified beforehand by using a fixed random seed. The quality of the individual is determined as an average quality over all simulation runs. The quality function consists of driven distance, fleet size, and out of stock situations as detailed in Section 3.4.2.

This reinforcement learning process was carried out for different scenarios leading to a diverse set of parameter settings. The aim was to analyze different exogenous and endogenous influence factors. Details on the scenarios can be found in Section 6.3 as well as in Section 3.4.4. For each scenario, a separate parameter setting was evolved leading to a policy that is adapted to the characteristics.

The individual parameter settings are illustrated in Table 4.1. In terms of interpretation of the parameters, it can be concluded that a replenishment strategy with rather small deliveries is used based on a high $Refill - Threshold$ and $RefillBarrier$ and a small $RefillFactor$ over all scenarios. In general, the interpretation of the evolved parameters is a difficult issue since they seem to span over the whole value range of the parameters depending on the scenario.

4.2.3 Investigation of the Parameter Landscape

To gain more insight about the characteristics of the parameter space, different local optima are analyzed for the 100% VMI scenario in terms of solution quality and Euclidean distance. For that purpose, a local search was executed with a maximum of 20 iterations and multiple restarts to obtain locally optimal parameter vectors. Seven different parameter vectors are presented in Table 4.2 and compared to the best found solution listed in the previous section (see Table 4.1).

Parameters	50 % VMI	100 % VMI	95 % SQ	90 % SQ
CapacityUtilization	1	0.47	0.99	0.46
PriorityThreshold	0.07	0.33	0.02	0.08
a_1 (MinOOS)	-0.37	-0.76	-0.32	-0.95
a_2 (AvgOOS)	-0.05	-1	-0.83	-0.17
a_3 (LastDelivery)	0.96	0.82	0.95	0.82
a_4 (InventorySize)	0.32	0.4	0.94	0.57
a_5 (Detour)	-0.83	-1	-0.81	-0.99
a_6 (Isolation)	0.54	0.08	0.26	0.14
RefillThreshold	0.49	0.61	0.51	0.64
RefillBarrier	0.79	1	0.98	0.44
RefillFactor	0.35	0.32	0.3	0.08
Parameters	Products	Demand	Clusters	Cluster
CapacityUtilization	0.87	0.99	0.99	1
PriorityThreshold	0.64	0.02	0.02	0
a_1 (MinOOS)	-0.03	-0.29	-0.32	-0.88
a_2 (AvgOOS)	-0.05	-0.71	-0.83	-0.26
a_3 (LastDelivery)	0.09	0.55	0.95	0.24
a_4 (InventorySize)	0.66	0.39	0.94	0
a_5 (Detour)	-0.26	0	-0.81	0
a_6 (Isolation)	0.24	0.86	0.26	0.8
RefillThreshold	0.53	0.53	0.51	1
RefillBarrier	0.99	0.98	0.98	1
RefillFactor	0.01	0.29	0.3	0.29

Table 4.1: Evolved parameters for the different VMI scenarios (cf. Vonolfen et al. (2013a))

Parameters	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7
Cap.Util.	1.00	0.58	0.61	0.72	0.98	0.92	0.99
PriorityThr.	0.00	0.00	0.67	0.03	0.27	0.12	0.44
a_1	-0.95	0.00	-0.17	-0.76	-0.91	-1.00	-1.00
a_2	-0.46	-0.55	-0.92	-0.43	-0.16	-1.00	-0.36
a_3	0.18	0.67	0.75	0.01	0.75	0.05	0.91
a_4	0.28	0.12	0.76	0.00	1.00	1.00	0.49
a_5	-0.02	0.00	-0.67	-0.35	-0.83	-0.31	-0.88
a_6	0.00	0.00	0.67	0.16	0.63	0.00	0.42
RefillThr.	0.47	0.42	0.33	0.92	0.66	0.58	0.45
RefillBarrier	0.64	1.00	0.89	0.92	1.00	0.69	0.56
RefillFactor	0.71	0.22	0.27	0.23	0.17	0.49	0.44
Rel. Quality	6.37%	5.70%	10.60%	2.72%	8.96%	5.80%	6.98%
Distance	1.55	1.43	1.08	1.36	1.31	1.36	1.06

Table 4.2: Different local optima for the 100% VMI senario. The quality (in the training phase) and the parameter vector (in terms of Euclidean distance) are compared to the best found parametrization for this scenario (which has been presented in Table 4.1).

The local optima have an average distance of 1.31 to the best found parametrization. No positive correlation between the distance and the relative quality compared to the best solution can be observed. This observation indicates a multi-modal fitness landscape with no globally convex structure.

	Run1	Run2	Run3	Run4	Run5	Run6
Run 2	1.31	-	-	-	-	-
Run 3	1.77	1.40	-	-	-	-
Run 4	0.94	1.22	1.69	-	-	-
Run 5	1.58	1.77	1.28	1.51	-	-
Run 6	1.00	1.66	1.54	1.30	1.45	-
Run 7	1.34	1.67	1.25	1.45	0.85	1.44

Table 4.3: Euclidean distance between the different local optima for the 100% VMI scenario which are listed in Table 4.2.

The distances between the individual local optima are listed in Table 4.3. The average distance between them is 1.4 while the average difference in solution quality is 2.99%. A weak positive linear correlation (0.28) exists between their distance and their difference in solution quality. However, parameter settings with similar solution quality that have a large distance between each other exist (for example Run2 and Run6).

4.2.4 Conclusions

Simulation-based evolutionary policy search was applied to a high-dimensional stochastic inventory routing problem characterized by an inherent problem complexity in terms of modeling and also in terms of computation time. Several approximations were combined to a heuristic approach to solve instances derived from real-world data and draw conclusions about endogenous and exogenous influence factors by means of evaluating the optimization potential of different scenarios.

The basis for the investigations was a simulation model containing the governing product consumption variables. The simulation was coupled with an approximate inventory replenishment and routing policy which was derived from domain knowledge. It has a fixed structure and was specifically designed to mitigate the issue of fluctuating demand patterns and achieve a constant resource usage. The policy was adapted to different scenario characteristics by evolutionary policy search in the space of parameter vectors.

Core of the approach were two sub-policies that make decisions which customer to replenish and what products respectively. They built on domain-specific features and reduced the problem complexity by abstracting from the state space allowing consideration of a large number of customers and products. They were adapted to scenario characteristics by setting predefined parameters. The parameter landscape has been investigated indicating globally non-convex and multi-modal characteristics with rather low fitness-distance correlation.

The presented methodology is an example for a policy that builds on domain knowledge and combines several approximations. The two-stage approach divides the inventory replenishment and routing decisions while estimating the effects on the quality of routes during the first phase. In a rolling-horizon fashion, only a single day of operations is optimized while considering a deterministic forecast of product consumption. These two measures aim at mitigating the problem complexity while still considering interactions between sub-problems and long-term effects. The policy combines several approximations and heuristics to allow the consideration of instances derived from real-world data. Using the simulation of different scenarios, the optimization potential could be evaluated considering various influence factors.

The study clearly shows the potential of the application of the methodology to intractable problems as a scenario technique overcoming the curses of dimensionality. However, it remains an open issue how close the performance of the policies get to a planning approach for computationally tractable smaller instances.

4.3 Priority Policies for Dial-a-Ride Problems

Apart from planning approaches based on static algorithms, routing policies have been applied to solve dynamic vehicle routing problems. The vehicles are considered as servers and instead of planning a routing sequence, the request to be served next is chosen ad-hoc according to a policy function.

The aim of this work is to generate the agent function (priority policy) by means of a simulation model and evolutionary policy search. The motivation stems from previous work performed in that area. Beham et al. (2009b) illustrated the potential of applying direct policy search to dynamic routing problems using policies with a fixed structure in the form of a parameter vector. The potential of evolving policies with a flexible structure by means of genetic programming was illustrated by van Lon et al. (2012). However, they did not use as many features as Beham et al. (2009b) to synthesize the policies and no comparison was performed neither between the approaches nor with planning algorithms. Combining these two approaches, Vonolfen et al. (2013c) illustrated the additional potential of policies with a flexible structure built on an extensive set of features. The performance of the evolved policies is compared in terms of solution quality and runtime to planning algorithms on dynamic dial-a-ride problem instances.

This section is based on and paraphrases parts of a previously published study of Vonolfen et al. (2013c) which was an extension of the methodology presented by Beham et al. (2009b) and van Lon et al. (2012). It uses the genetic programming implementation in HeuristicLab proposed by Kommenda et al. (2012). The main contribution in the context of this thesis are the consideration of feature selection in the direct policy search process and the evaluation of different policy representations in comparison with planning algorithms.

4.3.1 Problem Definition

The dial-a-ride problem (DARP) is concerned with transporting people from sources to destinations using a fleet of vehicles having practical applications ranging from door-to-door transportation to taxi-cabs. It can be regarded as a variant of pickup and delivery problems while the service quality in terms of maximizing user convenience is the main objective. In the dynamic variant, not all requests are known in advance but appear during the planning horizon (Berbeglia et al., 2010).

The dynamic DARP considered in this work is a shared taxi system where users can dynamically place requests between predefined service points. A homogeneous fleet of taxi buses, which can carry up multiple people at once,

serves the requests. The ability to carry multiple people at once enables synergy effects to be utilized between the transported requests while having to consider the user convenience for each individual passenger.

The requests arrive at predefined service points which are uniformly distributed in an Euclidean plane. For each request, an origin and target location is given as well as the number of people to be transported which determines the required service time. The user convenience is modeled as the lead time between appearance of the request at the source location and its delivery to the target location.

$$\min \sum_{i=1}^{|R|} l_{r_i} + \exp(\max(0, l_{r_i} - b)) \tag{4.6}$$

The objective function to be optimized is listed in Equation 4.6 (cf. Vonolfen et al. (2013c)). The aim is to minimize the total lead time over the set of requests R. For each request $r_i \in R$ its lead time is given by l_{r_i} and is defined as the time between its appearance at the source and its arrival at the target location. To avoid the starvation of individual requests, a maximum lead time bound b is defined in which each request has to be serviced. If the lead time of a request exceeds that limit, an exponential penalty is applied for the tardiness.

Assuming each vehicle (taxi-bus) as an autonomous entity, the problem can be modeled as a multi-agent system. In contrast to planning a set of routes using a central decision support system, multi-agent systems apply a decentralized decision making process. Each vehicle makes its own decisions according to an agent function with a possible interaction with the other agents.

The agent function can be represented as a dynamic policy who prioritizes the requests and selects the request with the highest priority to be served next. Each agent individually executes the policy to decide what action to perform next after the previous action has been completed. The decision is made ad-hoc and no planning ahead occurs. If multiple agents chose the same action at the same time, the agent where the policy returned the highest priority value performs the action.

4.3.2 Evolutionary Generation of Priority Policies

The proposed priority policy is based on several domain-specific features which were presented by Beham et al. (2009b) and extended by Vonolfen et al. (2013c). Basically, four aspects of the current system state are considered to determine the priority individually of each pending request: request

properties, location properties, other requests already picked up by the vehicle, and other agents.

Information about each **waiting request** includes the demand and temporal properties with the following features:

- *DemandSize* - Number of people to be transported between the two specified locations
- *StartDate* - Time the request arrived in the system
- *DueDate* - Time the request is due (with respect to the maximum lead time)
- *LeadTime* - Lead time of the request if it would by served immediately including the driving and service times

Requests appear at predefined pickup and delivery locations. When evaluating a request, the following features about its **location** are considered:

- *Distance* - Distance to the current agent
- *Min/Avg/Max-DistanceToDestinations* - Minimum, average, and maximum distance to other locations (remoteness)
- *EarliestTimeOfArrival* - Earliest arrival time of the agent at the location
- *PickupOrdersAtTarget* - Number of pickup requests waiting at that location
- *PickupOrderItemsAtTarget* - Total pickup demand waiting at that location which is the sum of the demands of all waiting requests

Since the vehicles can carry multiple customers, features regarding **carried requests** already picked up by the current agent are considered:

- *DeliveryOrdersAtTarget* - Requests to be delivered to that that location by the current agent
- *DeliveryOrderItemsAtTarget* - Total demand waiting to be delivered to that location which is the sum of the demand of all picked up requests
- *EarliestDueDate (EDD)* - Minimum due date of any carried request heading to the same target as the location of the waiting request

Since the decision making is decentralized, self-organizing behavior between the agents is required based on the following features regarding **other agents**:

- *Min/Avg/Max-DistanceToOtherCouriers* - Minimum, average, and maximum distance to the other agents
- *NumberOfOtherCouriersToTarget* - Number of agents heading to the same location (agents with a common target)

In total, 18 different features are used considering information about waiting and carried requests, locations, and other agents. All features are normalized in the range $[-1, 1]$ and combined to a priority value for each request by means of a priority policy.

Direct policy search is applied to generate priority policies based on the features for two different scenarios (details on the test environment can be found in Section 6.4). In the first scenario, an average number of 4 requests (low traffic intensity) while in the second scenario, an average number of 6 requests (high traffic intensity) appears per minute. For each scenario, a training set of 7 instances was used.

Previous research has shown, that for different traffic intensities different priority rules are needed as outlined in Section 2.2.1. For the generation of specialized policies for the two scenarios with low and high intensity, two different methods for direct policy search are examined. On the one hand, a policy with a fixed structure in the form of a linear representation is used and a parameter vector is evolved. On the other hand, a policy with a flexible structure is evolved represented as a formula tree.

Linear Representation

The linear representation of the priority policy builds a weighted sum of the 18 normalized features as listed in Equation 4.7. The policy has 18 parameters (a_i, where $1 \leq i \leq 18$) which represent the weights of the individual features.

$$\lambda_n = \sum_{i=1}^{18} f_{ni} * a_i \tag{4.7}$$

The policy search is carried out on a real-valued vector using an evolution strategy. Each element of the vector represents a weight in the range $[-1, 1]$. In terms of parametrization, a population size of $\mu = 4$ and an offspring size of $\sigma = 16$ was used in combination with comma selection. A self-adaptive

Gaussian mutation was used with learning parameters $\tau = 0.4$ and $\tau_0 = 0.4$. In addition, a heuristic contious crossover was applied. The process was repeated for 200 generations where 3204 evaluations were carried out in total.

Feature	Weight
EarliestTimeOfArrival	-0.94
EDD	-0.89
Distance	-0.89
MinimumDistanceToDestinations	-0.61
LeadTime	-0.52
MaximumDistanceToOtherCouriers	-0.47
DemandSize	-0.26
PickupOrdersAtTarget	-0.15
MaximumDistanceToDestinations	-0.06
AverageDistanceToDestinations	-0.06
DeliveryOrderItemsAtTarget	0.02
DueDate	0.06
StartDate	0.26
PickupOrderItemsAtTarget	0.31
NumberOfOtherCouriersToTarget	0.32
AverageDistanceToOtherCouriers	0.94
DeliveryOrdersAtTarget	1.00
MinimumDistanceToOtherCouriers	1.00

Table 4.4: Evolved parameters for the low intensity scenario ordered by their weight

The best found parameters for the low intensity scenario are listed in Table 4.4. According to their weight, the most impact aspects are the distance of the location to the vehicle ($EarliestTimeOfArrival$, $Distance$) as well as the waiting orders at that location ($DeliveryOrdersAtTarget$), the due date of the carried requests (EDD), and self-organizing aspects between the vehicles ($DistanceToOtherCarriers$).

For the high intensity scenario, the best found parameters are listed in Table 4.5. Similar to the low intensity scenario, self-organizing aspects between the vehicles, the distance of the location to the vehicle, as well as the number of waiting orders at a location are important factors. However, the due date of the request is a more important factor compared to the low-intensity parameters.

Feature	Weights
EarliestTimeOfArrival	-0.95
AverageDistanceToDestinations	-0.91
MaximumDistanceToOtherCouriers	-0.84
Distance	-0.51
LeadTime	-0.45
DeliveryOrdersAtTarget	-0.15
AverageDistanceToOtherCouriers	-0.13
DeliveryOrderItemsAtTarget	-0.09
EDD	-0.04
MinimumDistanceToOtherCouriers	-0.03
MaximumDistanceToDestinations	0.01
MinimumDistanceToDestinations	0.1
NumberOfOtherCouriersToTarget	0.37
PickupOrderItemsAtTarget	0.44
StartDate	0.5
PickupOrdersAtTarget	0.7
DueDate	0.77
DemandSize	0.79

Table 4.5: Evolved parameters for the high intensity scenario ordered by their weight

Tree Representation

A tree representation provides a representation for evolving priority policy formulas without assuming a structure a-priori. In contrast, the linear representation uses a fixed model and exposes a set of parameters which are optimized during the policy search. The tree representation seeks to overcome these limitations by capturing non-linearities and synthesizing a priority function based on the features without making a-priori assumptions about the model structure. Genetic programming (GP) is an established method for evolving tree structures as solutions for structurally complex problems and is applied for finding priority policies represented as trees in this context.

While for the linear representation the solution space is defined in the space of parameter vectors, it is defined as a set of trees that follow a certain grammar for tree representations. The root symbol of a tree returns a numerical value which is calculated by combining arithmetic (addition, subtraction, multiplication, division, average, exponential, logarithm), logical (if, greater, less, and, or, not), and terminal symbols (constants and features). The result of the tree evaluation is interpreted as the priority value of a given request.

In particular, genetic programming with offspring selection (Affenzeller et al., 2009) is applied which is an enhanced version of the standard GP algorithms and aims at the preservation of relevant genetic information. The parametrization of the algorithm was borrowed from the work of Pitzer et al. (2011) who applied a similar methodology to production logistics. A population size of 100 with a mutation rate of 15% and 1-elitism is used. The offspring selection is parametrized with a maximum selection pressure of 200, a comparison factor of 1, and a success ratio of 90%.

The comparably low population size for GP standards stems from the computational complexity of the simulation evaluation. To avoid over-adaption, the complexity of the trees was limited to a depth of 15 and a length limit of 25. The evolutionary process required 29072 evaluations in total to generate the priority policies.

The evolved priority policy tree for the low intensity scenario is illustrated in Figure 4.3 while the tree for the high intensity scenario is illustrated in Figure 4.4. The trees combine selected features to a formula that calculates the priority value by using logical and arithmetic expressions.

The interpretation of the relevant features is done from a more global perspective compared to the linear model. The larger population size of the genetic programming process in combination with the implicit selection of the relevant features allows an analysis of the variable frequency in the final population. The methodology of analyzing the variable impact is detailed by Kronberger (2011).

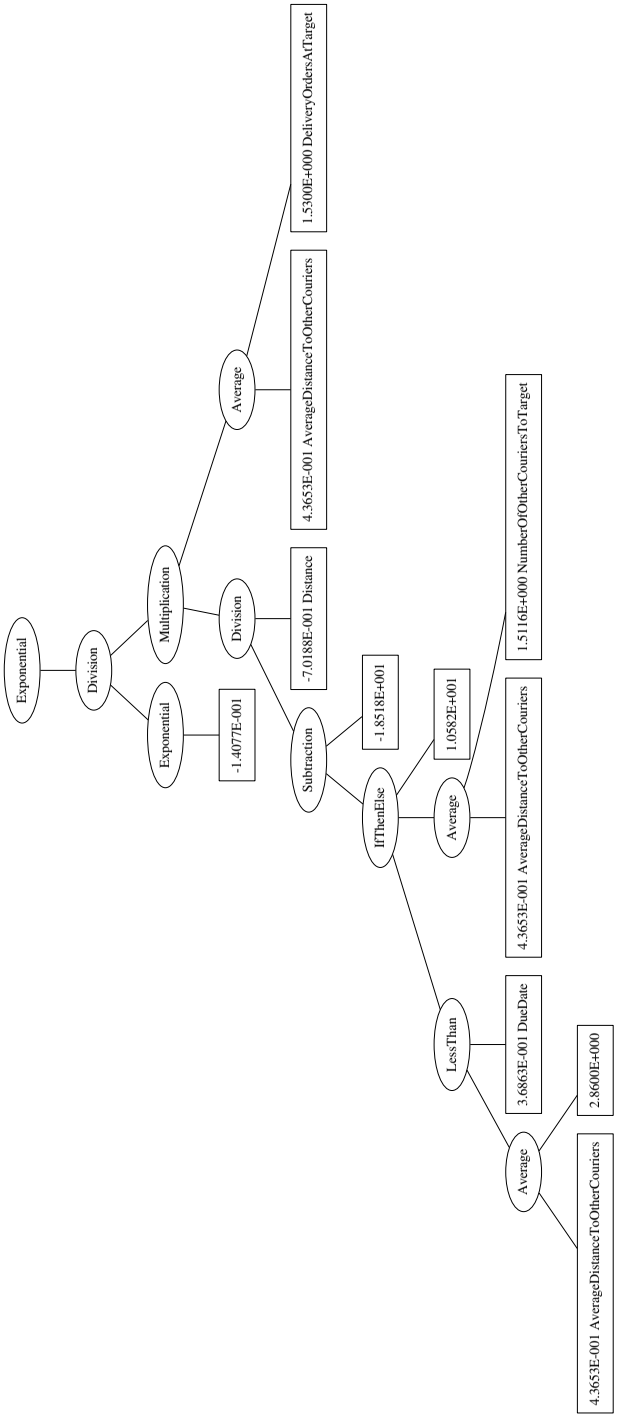

Figure 4.3: Evolved priority policy tree for the low intensity scenario

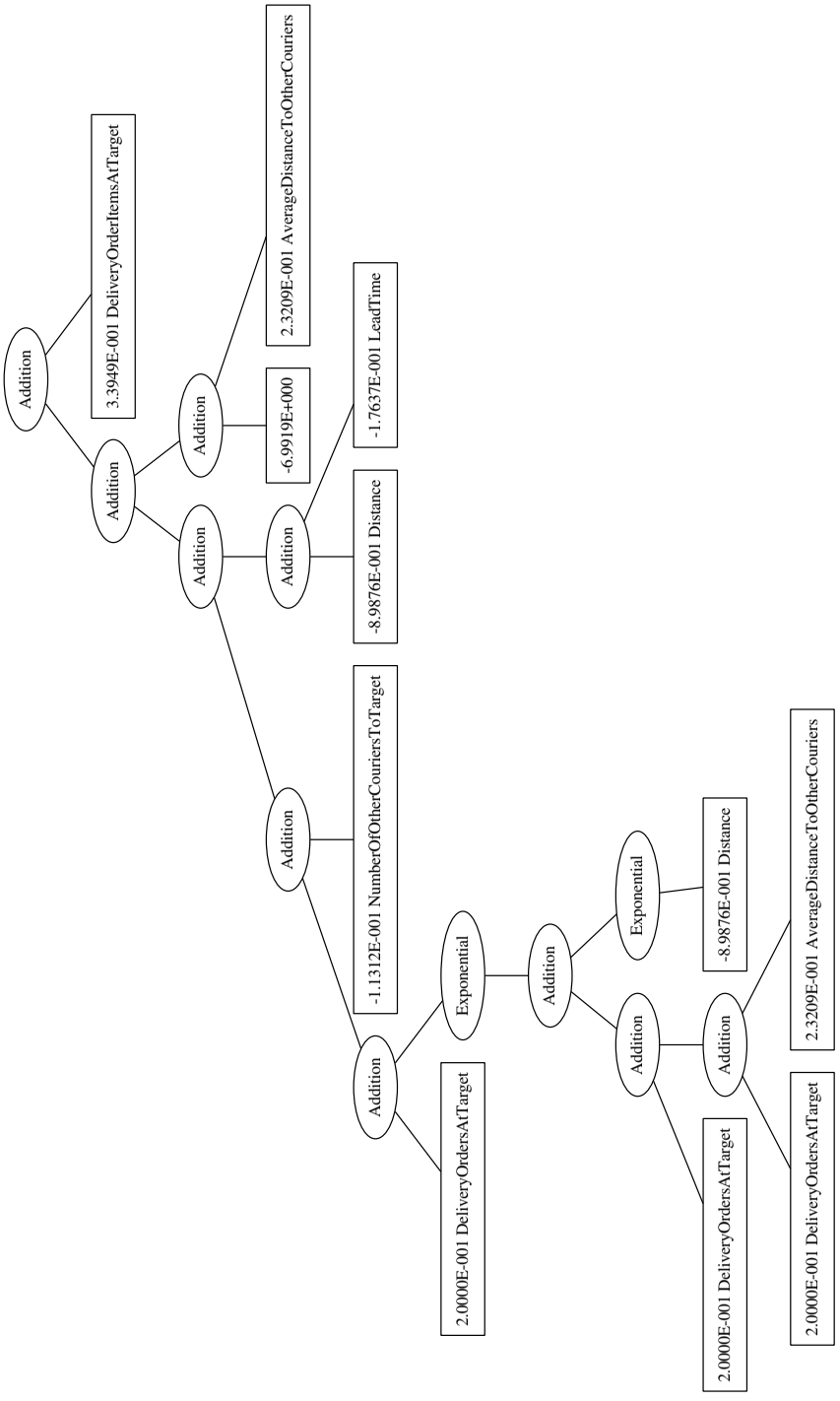

Figure 4.4: Evolved priority policy tree for the high intensity scenario

The variable frequency analysis for the low intensity scenario is depicted in Figure 4.5 and for the high intensity scenario in Figure 4.6. Important aspects for the determination of the request priority are the distance to the vehicle ($Distance$), the number of waiting requests at a location ($DeliveryOrdersAtTarget$), vehicle self-organization ($DistanceToOther - Couriers$), as well as temporal properties of a request indicating its urgency ($LeadTime$, $DueDate$).

These results correspond to the interpretation of the linear models while a more clearer picture is drawn due to an implicit feature selection in the GP process.

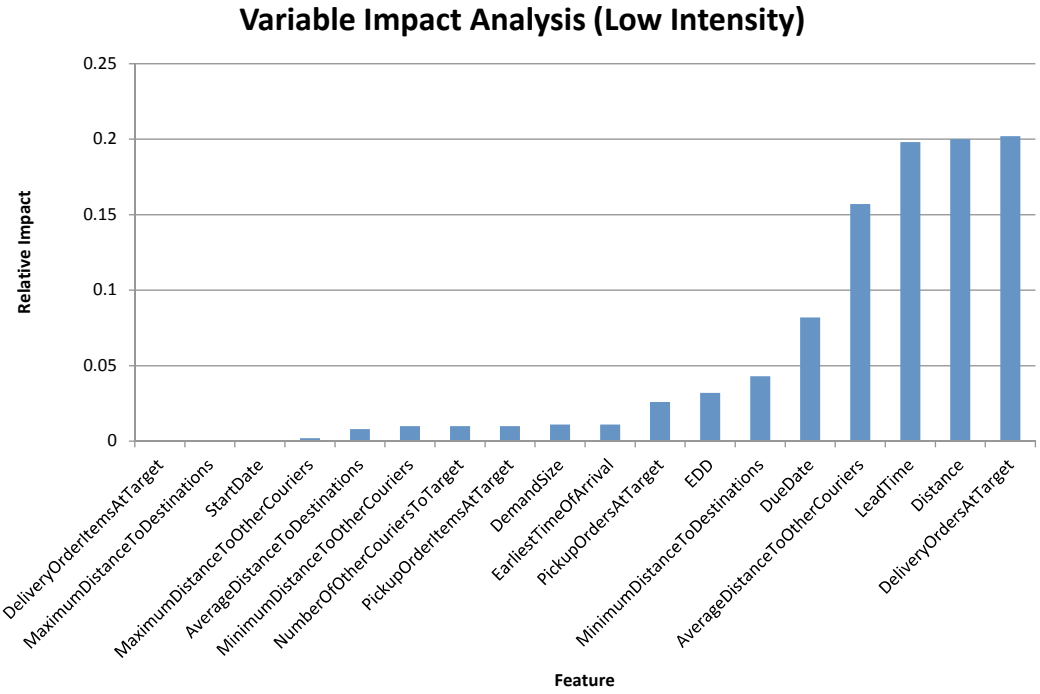

Figure 4.5: Variable impact analysis for the low intensity scenario

4.3.3 Conclusions

Conclusions drawn from the experimental setup and numerical results presented in Section 6.4 are twofold and concern the interpretation of the evolved policies on the one hand and the evaluation in terms of runtime and solution quality on the other hand.

In terms of interpretation, the clearest picture can be drawn from the genetic programming process due to its implicit feature selection. The three dominant aspects are identified based on the evolution of the policies from a

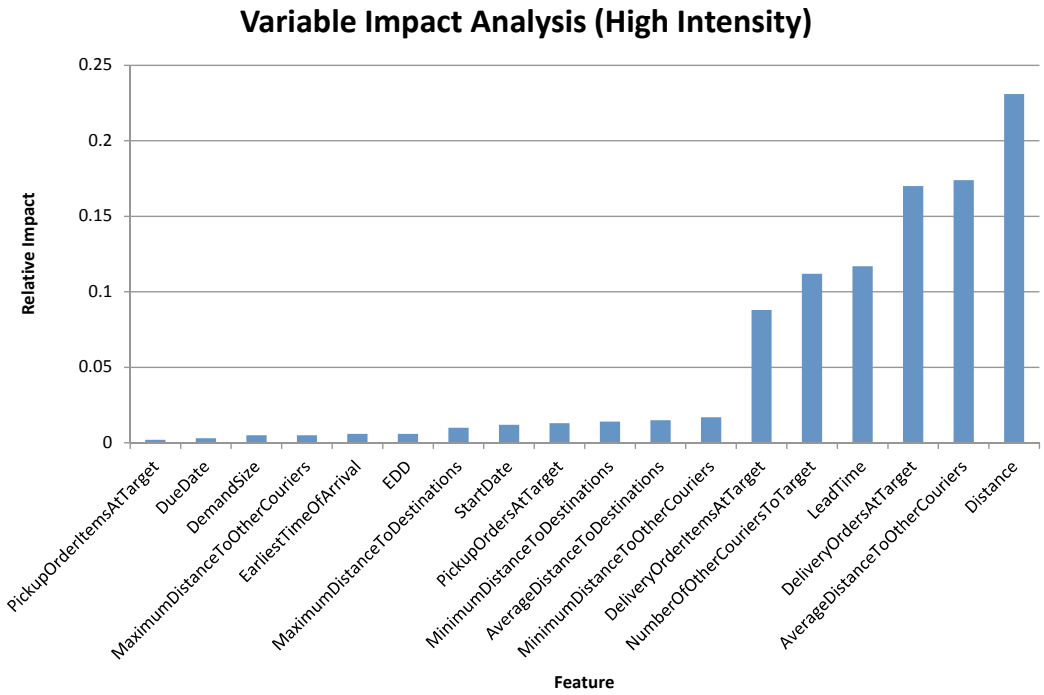

Figure 4.6: Variable impact analysis for the high intensity scenario

comparably large set of domain features (cf. Vonolfen et al. (2013c)). Firstly, an important aspect is related to the service quality determined mainly by the lead time of a request and its urgency. Secondly, the current effort to service a request for a vehicle is important w.r.t. distance and synergy effects with already picked up requests. Thirdly, self-organizing aspects between the agents are beneficial to split up the geographic space between the vehicles.

In terms of the achieved results on the test set, both the evolved linear as well as tree policies clearly outperform simple dispatching policies in both scenarios while having a comparable computational complexity. The computational effort is shifted to the training phase while the tree representation requires significantly more evaluations due to the more complex solution space. When applied online after the training phase, the evolved policies have a low computational complexity which is comparable to simple policies such as FCFS.

Additionally, the policies were compared to a planning algorithm. In particular, an online tabu search was used which re-optimizes a planned set of routes each time a new request appears. Unsurprisingly, the tabu search algorithm has a much higher computational complexity than the policies.

For the low intensity scenario, there was a gap of 4.76% for the linear and a 3% gap for the tree policy compared to the tabu search. The tree dispatching

policy yielded a significantly better result than the linear policy. In terms of runtime, the policies required a under 1 minute of total computation time while the tabu search required over 50 minutes.

In terms of the high intensity scenario, the gap to the planning algorithm decreased. Both the linear (gap 0.69%) and the tree (gap -0.34%) achieved comparable results to the planning algorithm and could be statistically not distinguished from each other. The tabu search had an average total computation time of over 60 minutes while the tree and linear dispatching policies required under 1 minute.

The results indicate, that specialized dispatching policies are applicable in highly dynamic environments where a rather reactive acting is required as opposed to planning a sequence of steps or when the computation time is a critical aspect. The computation complexity is shifted to a training phase and the evolved policies can be efficiently applied in an online setting with a comparable runtime to simple policies while performing significantly better.

However, a large gap can be observed between the static and the dynamic problem variant. Assuming that all requests would be known in advance, a static tabu-search algorithm can achieve over 30% better quality for the low intensity and nearly 15% better quality of the high intensity scenario compared to the online tabu search. The optimization potential reduces with increasing traffic intensity since the adherence to the maximal lead time becomes a more dominant factor and the problem is more constrained. In general, there is a large potential for incorporating a-priori information about the appearing requests to close the gap between the static and the dynamic problem variant.

4.4 Waiting Policies for Pickup and Delivery Problems

The anticipation of future requests is an important aspect of dynamic vehicle routing problems where requests appear during the planning period. Waiting policies determine strategically beneficial locations for vehicles to wait where new requests are likely to appear to utilize synergy effects.

Research on the distribution of waiting time along the vehicle routes has led to general heuristics on the one hand and waiting policies that incorporate knowledge about future request patterns on the other hand. In previous work, non-robust behavior of waiting policies depending on problem characteristics has been observed requiring specialized policies with adjusted parameter settings (cf. Ichoua et al. (2006)).

The aim of this work is to generate specialized waiting policies for pickup and delivery problems with various spatial and temporal characteristics. Two different ways to incorporate information about future requests are examined. The first way is to utilize stochastic demand distributions as proposed by Ichoua et al. (2006) while the second way is a newly proposed intensity measure. The evolved policies are evaluated in terms of robustness with respect to different problem characteristics and compared to general waiting heuristics that do not take into account information about future requests.

4.4.1 Problem Definition

The different waiting policies have been proposed for and evaluated on different variants of dynamic vehicle routing problems. To provide a general test bed, the different waiting policies are evaluated on pickup and delivery problems with time windows (PDPTW) in this work. Applications of the dynamic PDPTW are manifold and include full-truckload problems, less-than-truckload problems and passenger transportation.

A fleet of vehicles has to serve a set transportation requests during a planning period (i.e. a day of operation). A request has to be fulfilled by exactly one service of a single vehicle, this means that split deliveries are not allowed. The routes always start and end at the depot and the capacity restrictions of the vehicles have to be considered as well as the time windows for pickup and delivery locations. In the dynamic formulation, not all requests are known in advance but are revealed during the planning period. A service request s is revealed at a certain time r_s and contains the pickup location l_s^p and the delivery location l_s^d. Each location l can be serviced in the time window defined by the opening time o_l and closing time c_l.

The objective is to minimize the required fleet size as well as the total driven distance. The costs are calculated as a weighted sum of the utilized vehicles and the driven distance. The costs for using a vehicle are set to $c_v = 3000$, the costs for the traveled distance is set to $c_d = 1$. According to this parametrization of the evaluation function, the main objective is to minimize the fleet size, the secondary objective to minimize the driven distance over a given planning horizon.

The waiting policies considered in this work distribute the waiting time a-posteriori after the routes have been determined based on the current requests. This means, that in the first step the routing is performed and then the scheduling where the vehicles should wait along the routes in the second step.

After servicing each request, the vehicle can decide to wait at the current location R_i or to move to the next location R_{i+1} in the route R. The decision

how long to wait at a current location is made after the service at R_i has been completed. The time how long the vehicle can wait until the route would become infeasible is determined by the time windows along the route. The slack thus is defined as the maximum time the vehicle can wait at position R_i in a route R until the route would become infeasible.

4.4.2 Distribution of Waiting Time

The aim of waiting policies is to distribute the waiting time along the route in such a way, that newly arriving requests can be incorporated efficiently. For instance, it makes sense to wait at a busy area where new requests are likely to appear. Utilizing these synergy effects minimizes the distribution effort leading to a lower fleet size and driven distance.

The investigated waiting policies can be categorized into general policies and policies that consider knowledge about future requests. The motivation to incorporate knowledge about demand patterns comes from the observation that human dispatchers typically utilize experience about intense geographical regions or peak times in the planning process. The integration of knowledge is investigated in the form of stochastic information as well by means of an intensity measure. The knowledge has been extracted from a set of training instances representing historical data as detailed in Section 6.5.

General Policies

General policies utilize universal heuristics to distribute the waiting time. The most straightforward waiting heuristics are the *DriveFirst* and *WaitFirst* strategies which reflect the two most extreme situations. The former strategy is never to wait and always leave immediately for the next location while the latter is always to wait at the current location while feasible. These trivial strategies provide a bound for comparison with more sophisticated approaches.

Two more sophisticated strategies were defined and investigated for the PDPTW by Mitrovic-Minic and Laporte (2004). In the *Dynamic* waiting strategy (DW), the route is partitioned into zones and each service zone contains a number of consecutive locations that are close to each other. Within a service zone, the *DriveFirst* strategy is used and between zones the *WaitFirst* strategy is applied. A variation is the advanced dynamic waiting (ADW) strategy where the time is distributed proportionally to the time span spent in the zone.

Several waiting heuristics for the DVRP were proposed by Branke et al. (2005). The *Depot* waiting strategy waits at the depot as long as it is feasible before starting a new tour. The *MaxDistance* strategy waits at the location with maximum distance from the depot as long as possible. The *Location* strategy distributes the slack time equally over all locations of a tour while the *Distance* strategy distributes it proportionally to the driven distance. The *Variable* waiting strategy was derived from an analytically proven optimal strategy for the single vehicle case. The vehicles do not wait until the time to return to the depot is equal to the slack time. After that, the remaining wait time is distributed proportional to the remaining distance.

Based on Stochastic Information

Ichoua et al. (2006) proposed a threshold-based heuristic that exploits probabilistic knowledge which will be denoted as the *Stochastic* waiting strategy. The waiting heuristic is based on a spatial and temporal separation of the service area into zones. For each zone, which is defined by a geographic area j at a time period m, an arrival rate is specified in terms of a Poisson parameter λ_{jm}.

Based on this information, the vehicle waits at the current location as long as it is feasible and not longer than a given maximum wait time Δ if the following conditions are satisfied:

- The distance to the next location is greater than a given threshold α
- The number of already waiting vehicles in the geographic zone is smaller than a given number V
- The probability for a request to occur during the waiting time in the vehicle neighborhood is greater than a given threshold s. The parameter β determines the size of the neighborhood.

The *Stochstic* policy has five parameters $(\Delta, \alpha, s, V, \beta)$ that are set according to problem properties. Ichoua et al. (2006) presented two hand-tuned parameter settings for low-intensity and high-intensity scenarios.

A data pre-processing step is required to generate the stochastic knowledge from the past request information. For that purpose the problem instances of the training set have been preprocessed by separating the requests into 225 geographic slices ($1 \leq j \leq 225$) and 15 time slices ($1 \leq m \leq 15$) leading to a total of 3375 zones for each problem class. For each zone, an arrival rate is specified in terms of a Poisson parameter λ_{jm}. The parameter λ_{jm} was calculated by averaging the number of arriving requests in that zone over all training instances.

Based on an Intensity Measure

After analyzing the strengths of existing approaches, three important factors have been identified that influence the decision how long a vehicle should wait at a given location. The aim is to distribute the waiting time in a fine-grained and proactive way.

The first factor is based on the observation that, if the next location is far away, it might be beneficial to wait. This has been considered by the partitioning approach used in the *Dynamic* strategy and the threshold for the distance used in the *Stochastic* strategy. The second factor is the relative intensity of the current location and to distribute the available slack proportionally. A proportional distribution has been successfully applied using the *Dynamic* and *Variable* strategies. The third factor is the available slack at a given time. The *Variable* waiting strategy proposes not to wait at the beginning of the route. This indicates, that the slack must be carefully distributed over the route.

Based on the strengths of previous waiting strategies, a new waiting policy is proposed that is based on an intensity measure which in the following will be denoted as the *Intensity* waiting strategy. The motivation to use an intensity measure instead of a stochastic model is motivated by the fact that stochastic information needs to be of a certain quality and might require intensive pre-processing steps in practice as pointed out by Ferrucci et al. (2012). The use of an intensity measure aims at removing these restrictions.

Instead of using a stochastic model, the intensity is calculated based on historical request data which has been for example collected during daily operations. In the case of this study, a set of training instances is used as historical data. The data consists of a set S of historical service requests where each service request $s \in S$ occurs in a certain planning horizon h. Each service request s is revealed at a certain time r_s and contains the pickup location l_s^p and the delivery location l_s^d.

The definition of the intensity measure is based on the transition time between two locations l_1 and l_2 at the current time t. The transition time is the time that passes between the time the vehicle leaves location l_1 and the time the vehicle starts servicing location l_2. It consists of the distance ($Dist$) between l_1 and l_2 and a possible waiting time before the time window at location l_2 opens. Formally it can be defined as listed in Equation 4.8.

$$TransitionTime(l_1, l_2, t) = Dist(l_1, l_2) + \max\{0, o_{l_2} - (t + Dist(l_1, l_2))\} \quad (4.8)$$

The intensity of a location l at the current time t is defined as the average transition time for requests in the historical request set S that would have

been not revealed yet and is listed in Equation 4.9. The transition times are normalized according to the total planning horizon h. Locations with a lower average transition time have a higher intensity.

$$Intensity(l,t) = 1 - \frac{\sum_{\{s \in S : r_s > t\}} \frac{TransitionTime(l, l_s^p, t)}{h}}{|\{s \in S : r_s > t\}|} \tag{4.9}$$

By combining these considerations into a single policy, the decision how long to wait at a current location R_i in a given route R or to move to the next location R_{i+1} after finishing the service at location R_i at time t_i is based on these three factors:

- The transition time between R_i and the next location R_{i+1} in route R. The transition time is a sum of the distance between R_i and R_{i+1} and the waiting time before the time window opens at R_{i+1}.
- The intensity of location R_i as opposed to location R_{i+1}.
- The slack in the route R at position R_i which is the maximum time the vehicle can wait at position R_i until the route would become infeasible.

To evaluate the benefit of waiting at a given location in the route, the three factors are combined into a single value. The formula for calculating the value of waiting in the given route R at a location R_i at time t_i is listed in Equation 4.10. The value of waiting is calculated at time t_i before moving to the next location R_{i+1} at which the service would be completed at time t_{i+1}. The function $Slack(R_i, t_i)$ calculates the maximum time at location R_i in a route R at time t_i that can be waited until a time window in the remaining section of the route would be violated. The transition time and slack are normalized according to the remaining time within the total planning horizon h after the location R_i has been served at time t_i.

$$\begin{aligned} v(R_i, t_i) &= \alpha' * \frac{TransitionTime(R_i, R_{i+1}, t_i)}{(h - t_i)} \\ &+ \beta' * \frac{Intensity(R_i, t_i)}{Intensity(R_i, t_i) + Intensity(R_{i+1}, t_{i+1})} \\ &+ \gamma' * \frac{Slack(R_i, t_i)}{(h - t_i)} \end{aligned} \tag{4.10}$$

Based on the value of waiting at a certain location the waiting policy which returns the amount of time the vehicle should wait at the current

position R_i of route R at time t_i is listed in Equation 4.11. The parameter ϵ' defines a threshold the value of waiting must exceed, otherwise the vehicle does not wait at the current location. The waiting time in the route is scaled according to the amount v exceeds the threshold. All features are normalized in the interval $[0, 1]$, thus the value v is in the range $[0, v_{max}]$. The maximal value of v is $v_{max} = \max\{0, \alpha'\} + \max\{0, \beta'\} + \max\{0, \gamma'\}$. For locations that have a high value of waiting, a large proportion of the available slack in the route is used.

$$WaitingTime(R_i, t_i) = \max\{0, Slack(R_i, t_i) * (\frac{v(R_i, t_i) - v_{max} * \epsilon'}{v_{max} - v_{max} * \epsilon'}\} \quad (4.11)$$

In total, the *Intensity* policy has four parameters of the waiting strategy that have to be set according to the problem characteristics: α', β', γ' and ϵ'.

4.4.3 Evolutionary Parametrization

Both waiting policies that incorporate knowledge about future demand patterns expose parameters that are set according to the problem characteristics to allow a fine-grained distribution of the waiting time. For example, in some scenarios the region intensity, in other scenarios the transition time might be more important. In Ichoua et al. (2006) the parameters have been set manually by means of preliminary experiments. In contrast, in this work, direct policy search is applied to automatically specialize the policies to different problem characteristics.

As stated earlier, the *Intensity* policy exposes five while the *Stochastic* policy exposes four parameters. The parameters are represented as a real-valued vector which is optimized using a self-adaptive evolution strategy. The parametrization is tailored to the computationally complex task of direct policy search. Thus, a comparatively small population size is used and the ratio between parents and generated children is high which leads to a greedy search process. Each generation, 16 children are created from the 2 parent individuals by heuristic recombination and a normally distributed mutation. The 2 best children replace the parents for the next generation (comma replacement). The mutation strength is adapted according to the learning parameters $\tau = 0.4$ and $\tau_0 = 0.4$. This process is repeated for 50 generations.

For the training phase a push forward insertion heuristic which has been adapted to the PDPTW by Li and Lim (2001) is applied for route calculation. It is characterized by a comparatively low computational complexity which is required since the policy search requires much computational resources.

The parameter vectors of the waiting policies are evolved for different problem classes, each containing a set of training and a set of test instances (details about the used benchmark instances can be found in Section 6.5). The classes are characterized by different spatial (C is clustered, R is random, RC is mixed) and time window (1 is tight, 2 is large) properties. The evolved parameter set for the different classed for the *Stochastic* policy are listed in Table 4.6 while the parameters for the *Intensity* policy are listed in Table 4.7.

Class	α	Δ	s	β	V
C1	1.000000	0.843837	0.206368	0.535048	0.000000
C2	0.136841	0.609929	0.575356	0.380122	0.336438
R1	0.000000	0.537976	0.604499	0.092322	0.753874
R2	1.000000	1.000000	0.999918	0.756439	0.948408
RC1	0.693591	0.459593	0.678374	1.000000	0.370727
RC2	0.000000	1.000000	0.793226	0.216212	0.429234

Table 4.6: Evolved Parameters for the *Stochastic* Policy

Class	α'	β'	γ'	ϵ'
C1	0.883440	0.519018	0.000000	0.000000
C2	0.000000	0.567995	0.275205	0.000000
R1	0.486998	0.080907	0.747825	0.104138
R2	0.000000	0.000000	0.673014	0.023979
RC1	0.942528	0.949774	0.165874	0.250547
RC2	0.053280	0.097974	0.800274	0.000000

Table 4.7: Evolved Parameters for the *Intensity* Policy

The fact that the evolved parameters are quite different for the individual classes indicates that specialized settings are needed depending on the problem characteristics. After the policies have been evolved on the training set, their performance is evaluated on the test set using a more sophisticated algorithm for route calculation. Concretely, the unified tabu search algorithm (Cordeau et al., 2002) is applied. The search process is based on a shift neighborhood where customers are moved from one tour to another using the best possible insertion position. The constraints are adaptively relaxed and tightened to be able to move through infeasible regions of the search space which makes the search process very powerful. In terms of parametrization, a tabu tenure of $5log_{10}n$ is used, where n is the number of customers. Every time a new request occurs, it is inserted at the best possible position of the current executing plan and 100 iterations are performed to calculate a new plan.

4.4.4 Conclusions

Based on the computation results presented in Section 6.5 the drawn conclusions concern the competitiveness of the evolved waiting policies in comparison to general heuristics as well as the influence of spatial and temporal problem characteristics to the potential of anticipatory waiting. The *Intensity* strategy had a slight advantage over the *Variable* strategy which was the best general heuristic, while the *Stochastic* strategy was clearly outperformed in the test phase. The average savings of the *Intensity* strategy on the overall test set were 4.59%.

The temporal and spatial characteristics have a large influence on the potential savings that can be achieved with anticipatory waiting. Especially geographically clustered customers are very beneficial where 11.71% savings could be achieved by the *Intensity* strategy compared to not applying a waiting heuristic. Also larger time windows are more beneficial than small time windows where over 7.14% savings were achieved. Also the degree of dynamism has an impact on the potential savings. The savings potential is largest for instances were around half of the customers appear dynamically with decreasing potential both for instances with higher and lower degree of dynamism. A possible explanation regarding the influence of dynamic requests is the fact that the lower the degree of dynamism is, the more requests are active at the same time and thus the waiting time can be distributed in a more fine-grained way. However, the gap to the static solution and thus the optimization potential of applying anticipatory waiting decreases.

It can be concluded that the best general heuristic, the *Variable* strategy, provides robust performance on the overall test set. It has been derived from a proven optimal policy in the single vehicle case and transfers well to the investigated problem variant. However, especially on instances with characteristics that are beneficial for applying waiting heuristics, additional optimization potential can be utilized by applying specialized waiting policies. On instances with clustered customers the evolved *Intensity* policies perform 1.35% better, on instances with large time windows they perform 1.11% better, and on instances with clustered customers as well as large time windows they perform 3.44% better than the *Variable* policy. On instances with mixed randomly and clustered geographically distributed customers as well as small time windows the picture is less clear.

Based on the results it can be concluded, that the potential of waiting heuristics in general and of specialized policies in particular strongly depends on the problem characteristics while specialized policies clearly can exploit additional optimization potential compared to general heuristics in cases where it is beneficial to apply anticipatory waiting.

Chapter 5

Dynamic Situational Selection of Routing Policies

In this chapter, a methodology for the situational selection of solution techniques within dynamic vehicle routing environments with changing characteristics is presented. The simulation optimization approach presented in Chapter 3 as well as the automated algorithm design by direct policy search presented in Chapter 4 are integrated in a methodological framework based on portfolio-based algorithm selection.

In Section 5.1 the different methodologies presented in this thesis are combined to a heuristic framework that is adaptive in terms of changing problem characteristics. Based on this framework, in Section 5.2 a case study is presented where a stochastic routing problem with changing uncertainty is solved using a portfolio of human as well as semi-automatically designed heuristics.

5.1 Portfolio-Based Dynamic Algorithm Selection

As pointed out in Chapter 2, research on heuristics for dynamic vehicle routing problems has mainly focused on techniques that are especially designed for certain problem characteristics on the one hand and on general search strategies that work well on a wide range of problems on the other hand. In many cases, specialized heuristics become suboptimal once the problem characteristics change. When designing general strategies, optimization potential is often lost due to a focus on average performance. This dilemma can be observed early in dynamic vehicle routing research when Bertsimas and Van Ryzin (1991) investigated policies discovering non-robust behavior.

These considerations are according to the *no free lunch* (NFL) theorem (Wolpert and Macready, 1997) which states, that there is no algorithm that performs best in all situations and there is a tradeoff between generalization and specialization. Focusing only on average-case performance of algorithms leads to the fact that strategies might be neglected that perform well only in certain situations while non-robust behavior was identified for highly specialized algorithmic strategies. An example are specialized waiting strategies investigated in Section 4.4 which do not show a large benefit in the average case, however improve the solution quality significantly for clustered problem instances with large time windows.

With these aspects in mind, it would be desirable to have a dynamic vehicle routing system that combines several specialized heuristics and adaptively selects an appropriate strategy according to the current problem characteristics. A methodological framework for a portfolio-based algorithm selection is presented in the following which combines elements from several streams of research and integrates the methodological developments presented in this thesis.

5.1.1 Methodological Framework for Algorithm Selection

Viewing the dynamic vehicle routing problem as a multi-stage optimization problem (as detailed in Chapter 2), a heuristic strategy is selected from a portfolio of specialized heuristics in each decision stage during the planning process reacting to a changing problem environment. At each decision stage, not only decisions about the routing are made on the problem level but also about the used routing policy that is suitable in terms of current problem characteristics on the meta level.

Core of the approach is a portfolio of specialized heuristics which can be human created or semi-automatically generated by direct policy search as outlined in Chapter 4. The assumption is, that combining different policies with individual strengths and weaknesses to a portfolio enables to overcome the dilemma of generalization versus specialization as presented by the NFL theorem. The aim is to create a portfolio that achieves better results than each policy individually. The portfolio can contain very specialized policies that probably only perform well under certain situations. By dynamically selecting a suitable policy from the portfolio, the methodology allows an adaption to changing problem characteristics and becomes more generally applicable than each individual policy.

The idea of combining several algorithms to a portfolio was first presented

by Huberman et al. (1997) and stems from applying principles of financing to optimization. Analogous to having a set of financial assets providing a return of investments, computational resources are invested by running multiple algorithms in parallel and the risk and reward in terms of solution quality is investigated.

As Silverthorn (2012) notes, modern portfolio approaches apply algorithm selection instead of the economical framework. Approaches based on machine-learning have been very competitive. Examples are the SATzilla system (Xu et al., 2008) in the area of satisfiability problems and the CP-Hydra system (O'Mahony et al., 2008) for constraint programming which won several competitions on international conferences. The research on portfolio-based algorithm selection is very active and an online survey is provided by Kotthoff (2014).

A theoretical model for algorithm selection was already presented by Rice (1976) and includes three important aspects. The problem space (P) is a set of considered problem instances. The algorithm space (A) is a set of algorithms that can be used to solve instances of the considered problem. The performance measure space (Y) includes possible criteria to measure algorithm performance such as runtime or accuracy. The aim is, for a particular problem instance $p \in P$, to find a selection mapping $S(p)$ that returns an algorithm $a \in A$ which maximizes the performance measure $y(a, p)$. Rice (1976) noted that problem and algorithm characteristics are important for algorithm selection as well as the definition of performance criteria.

Derived from the above definition, methodological approaches dealing with the algorithm selection problem must tackle three important aspects as outlined by Cruz-Reyes et al. (2012):

- Selection of *problem features* to characterize problem instances
- Selection of a set of *algorithms* with different strengths and weaknesses with respect to a large range of problem instances
- Creation of a *mapping* between the problem space and the algorithm space to select an algorithm for a given problem instance connecting problem features with algorithms

These three aspects are considered in the presented methodology which is illustrated in Figure 5.1. It is based on the simulation optimization model presented in Section 3.1 and extends it by replacing the single optimization component with a portfolio-based algorithm selection while the interaction with the simulation environment stays the same[1].

[1]For details on the world state representation, the interaction mechanisms, and the optimization model see Section 3.1

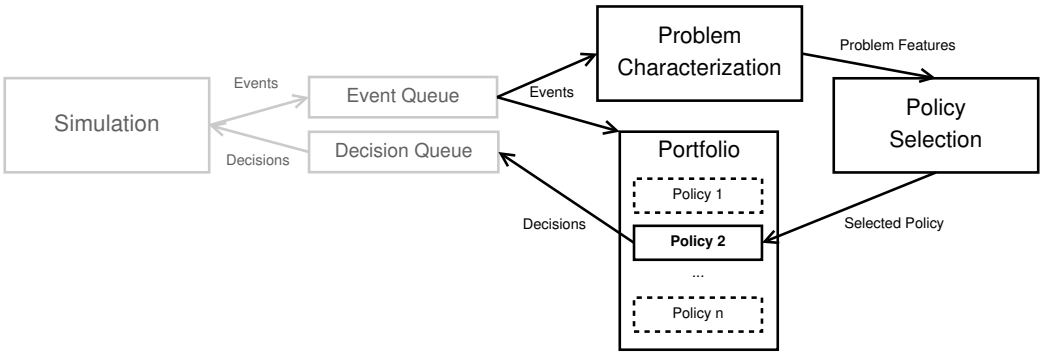

Figure 5.1: Methodological framework for dynamic algorithm selection (illustrated in black) extending the simulation optimization approach presented in Section 3.1 (illustrated in light gray)

Events received from the simulation indicate changes of the problem environment (e.g., a new order was placed) which are tracked by the *Problem-Characterization* component to update the current problem feature values as well as by each policy in the *Portfolio* to update its world state. The *PolicySelection* component selects a policy from the portfolio based on the current problem features. The selected policy is used to make routing decisions and can incorporate any of the solution methods presented in Section 2.2. In principle, a different policy can be selected whenever the problem environment changes allowing an adaption of the solution strategy to changing problem characteristics.

The methodological framework combines problem characterization, portfolio design, and problem to algorithm mapping. These three aspects will be examined in the following and the base framework will be extended to incorporate learning in terms of extending the portfolio and improving the mapping.

Problem Characterization

Problem characterization has the purpose of defining a set of features that differentiate problem instances and serves as a basis for algorithm selection. The importance of selecting appropriate problem features has been already pointed out when the algorithm selection problem was initially defined by Rice (1976). Recently it was highlighted by Cruz-Reyes et al. (2012) who noted that the definition of appropriate problem features is a difficult task and a meaningful problem differentiation is needed for building a mapping and understanding the connection between problem characteristics and algorithm performance.

As Cruz-Reyes et al. (2012) points out, there are two approaches to characterize problem instances. On the one hand, problem dependent features can be derived from domain knowledge while on the other hand a more general fitness landscape analysis can be performed. Both methods have been applied in the past to characterize vehicle routing problems.

In terms of problem specific features, the classical VRP has been studied with and without time windows. Solomon (1987) has pointed out the significance of spatial and temporal problem characteristics of VRPTW instances and proposed an often used benchmark set with different combinations of these. Random, clustered, and mixed geographically distributed customers were considered as well as small and large time windows. Six different classes of problem instances with different number of customers were generated combining these properties. Ruiz-Vanoye et al. (2008) proposed complexity indicators based on descriptive statistics that consider the geographical and demand distribution of the customers, the customer number, time window properties, and service time distributions. Pitzer et al. (2012) investigated a VRP without time windows and considered the problem size, the clustering as well as the geographical eccentricity of the customers, and distance as well as demand distributions.

In contrast to problem specific features, fitness landscape analysis measures are problem independent and thus more generally applicable. Assuming a certain neighborhood, different properties of the resulting fitness landscape are analyzed in terms of statistical as well as information theoretical measures. The analysis of fitness-distance correlation of CVRP instances revealed that many instances had a *big-valley* structure (Kubiak, 2007) which means that local optima are clustered in a globally convex structure. Ventresca et al. (2013) analyzed VRPTW instances with information theoretic landscape measures such as the ruggedness, neutrality, and the basin of attraction as well as the isolation of local optima. Pitzer et al. (2012) analyzed CVRP instances using several fitness analysis features that included both statistical and information theoretical measures.

As Pitzer et al. (2012) concluded, both problem dependent and fitness landscape analysis measures can provide a valuable contribution to drawing a complete picture of the problem instance characteristics and that a correlation exists between them. An important topic that can be investigated based on the problem features is the discrimination in terms of problem difficulty. Ventresca et al. (2013) used information theoretic landscape measures to cluster problem instances in terms of their difficulty. Also for other combinatorial optimization problems, such as the quadratic assignment problem (Pitzer et al., 2014), a correlation between problem features and its hardness for certain search methods has been found.

Portfolio Design

Portfolio design is concerned with selecting a set of algorithms and combining them together to solve a diverse set of instances for a given problem. As pointed out by Gomes and Selman (1997), creating a portfolio of algorithms makes sense in cases where the algorithms show different performance profiles and no single algorithm dominates the others on all problem instances. Cruz-Reyes et al. (2012) state that creating a portfolio of algoirthms is typically motivated by the fact that for a given problem usually a large range of solution techniques exist that have different strength and weaknesses. The portfolio can be designed beforehand (a-priori) or by means of online or offline learning.

In terms of an a-priori selection of the set of algorithms to be included in the portfolio, as noted by Cruz-Reyes et al. (2012), portfolios usually contain several state-of-the-art algorithms for a given problem. The selection of the set of algorithms to be included in the portfolio is typically based on a diverse set of problem instances on which the performance is evaluated empirically. When choosing a representative instance set which covers a wide range of problem characteristics for the selection step, the risk is minimized that the portfolio will perform poor on unseen problem instances.

By integrating automated algorithm design, the portfolio can be extended with newly generated algorithms. This approach was first studied by Xu et al. (2010) for satisfiability problems and showed advantages over pre-defined portfolios being particularly advantageous in problem domains that require highly specialized algorithms. The method requires an automated algorithm design procedure as well as a portfolio building technique.

Extending the portfolio by newly generated policies can be viewed as a learning capability to adapt to previously unseen problem characteristics. A methodology integrating the policy generation approach presented in Section 4.1 in the portfolio architecture is illustrated in Figure 5.2. An *ExplorationElement* explores previously unseen problem features (for example by reacting to problem features that are encountered during online operation). Based on these features, a *ProblemGeneration* component generates new problem data that is used to search for an efficient policy using direct policy search (see Section 4.1). The *PortfolioExtension* component can then decide to include these newly generated policies in the portfolio or to replace a given policy by a new one based on the performance measures.

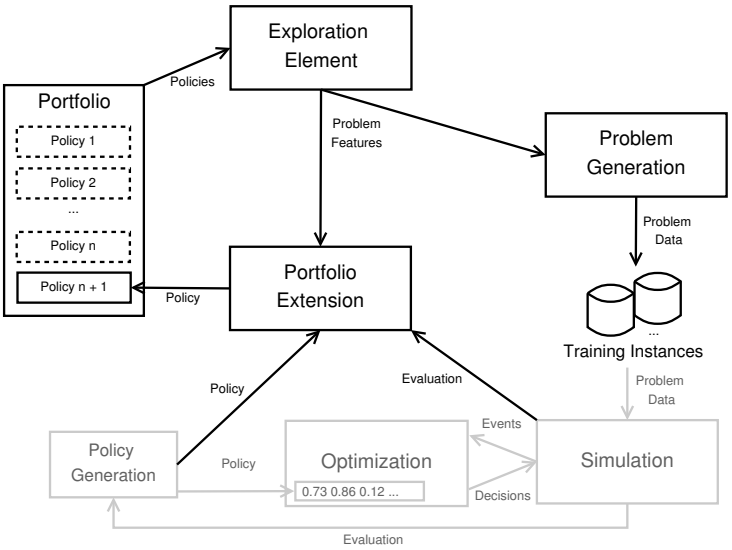

Figure 5.2: Methodology extending the portfolio with newly generated policies using the methodology presented in Section 4.1 (illustrated in light gray) by exploring new problem features (illustrated in black)

Problem to Algorithm Mapping

A problem to algorithm mapping mechanism is the core of a system for automated algorithm selection. The aim is to map from problem features to an algorithm that provides the best performance measure. In the case of dynamic vehicle routing, a routing policy is selected based on the current problem characteristics.

As Smith-Miles (2008) notes, many different research directions have been followed in several areas. In the area of meta-learning, machine learning methods have been investigated to learn models relating problem features with algorithm performance. Work in the area of artificial intelligence focused on empirical hardness estimations while within research on metaheuristics search spaces were characterized using fitness landscape analysis. A conclusion is that an intersection of techniques from multiple disciplines is required to provide an efficient mapping mechanism for algorithm selection. Several methods have been investigated ranging from human-defined heuristic rules to learning approaches based on meta-knowledge about algorithm performance.

A rather straightforward approach is the use of heuristic rules that select and algorithm based on problem features and are defined beforehand by a domain expert. For example, Beck and Freuder (2004) proposed different selection strategies for scheduling problems based on limited problem instance

knowledge with inexpensive metrics. Obvious disadvantages are the requirement for human experience in the design of the rules and the lack of learning capabilities once the rules have been defined.

The use of machine learning methods for the development of algorithm selection models was first studied in the context of selecting learning algorithms for classification problems according to Smith-Miles (2008) who generalized the idea of meta-learning to other application areas such as combinatorial optimization. Kotthoff et al. (2011) surveyed and compared various machine learning methods for algorithm selection:

- *Case-based reasoning* does not build a model or theory about algorithm performance but finds the examples of past performance (cases) to infer about unseen problem instances. For example, the nearest neighbors of a new problem instance could be found in the past performance data based on the problem features.

- *Classification* labels the different problem instances with an algorithm that should be used to solve them. A classifier is learned that discriminates the problem instances according to their features.

- *Regression* predicts the performance of each algorithm on a given problem instance. The algorithm with the best predicted performance can be selected.

- *Statistical relational learning* predicts performance ranking of algorithms for a particular problem instance while considering uncertainty. For example, support vector machines can be applied to predict the ranking scores. It is a rather new approach to algorithm selection.

The base framework for portfolio-based algorithm selection (presented in Figure 5.1) can be extended with machine learning techniques to learn about the problem to algorithm mapping as illustrated in Figure 5.3. A *PerformanceEvaluation* component provides feedback about the performance of applying a selected policy in a situation with certain problem characteristics. A *Reasoning* component generates mapping knowledge by applying machine learning techniques to build a meta-model explaining the policy performance based on the current set of problem features. That way, the system can continuously extend the mapping knowledge offline during training or online during operation and learn about the mapping between problem features and policy performance.

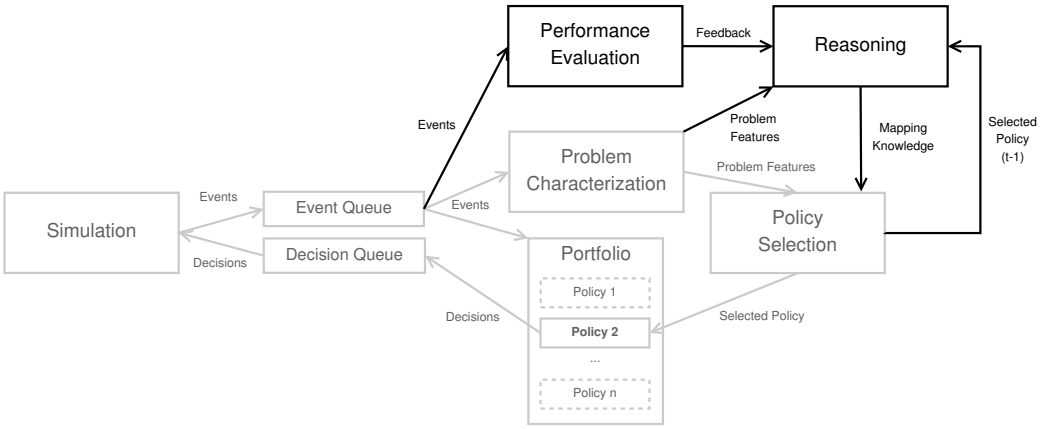

Figure 5.3: Methodology extending the base framework for algorithm selection (illustrated in light gray) with capabilities for learning mapping knowledge by a feedback and resoning component (illustrated in black)

5.1.2 Specializations and Possible Application Areas

The application area of the proposed framework is focused on dynamic vehicle routing problems where the characteristics change during operations requiring to change the solution method to utilize the full optimization potential.

In principle, the problem characteristics of a dynamic vehicle routing environment can change with any new information that arrives during operation (see Section 2.1.3. For example, newly arriving requests can change the overall spatial and temporal characteristics such as geographical distribution of the customers or time window properties. Also, the degree of dynamism might change during operations requiring to switch between reactive policies and planning algorithms. When stochastic information is available, the information quality might change requiring different solution approaches.

Depending on the application, specializations of the general methodological framework can be created. There are three important design decisions when specializing the general framework for portfolio-based algorithm selection. They can be summarized as following:

- The *problem characterization* can be achieved by problem dependent or general fitness landscape features. Also, a combination of both is possible.

- The *portfolio design* can be done either a-priori using a set of available pre-defined algorithms or by means of learning. The learning can occur offline on training instances as well as online during operations.

- In terms of *problem to algorithm mapping* human-defined heuristic rules or learning approaches can be used. The machine learning process can be carried out offline on training instances as well as online as new problem characteristics are encountered. When the algorithm portfolio is extended online, also the mapping must be adapted.

In the following, a specialization of the methodological framework for portfolio-based algorithm selection will be presented which is applied to a stochastic routing environment with changing uncertainty.

5.2 Combining Several Heuristics in Environments with Changing Uncertainty

In this section, a dynamic pickup and delivery problem with stochastic customers is investigated where the appearance probabilities of the customers change over time. The robustness behavior of various policies in terms of uncertainty of the stochastic knowledge on instances with different spatial and temporal characteristics is investigated. On the basis of the robustness analysis, the different policies are combined to a portfolio by specializing the portfolio-based algorithm selection framework presented in the previous section.

5.2.1 Context and Motivation

The incorporation of advance knowledge about future events allows a pro-active planning process and potentially improves the solution quality compared to purely reactive planning. However, previous studies have shown that the quality of the advance knowledge has a major impact on the potential savings. Thus it is important to consider the robustness of the pro-active planning processes in terms of data quality considering the temporal and spatial properties of the problem environment.

Bent and Van Hentenryck (2004b) noted that the incorporation of advance knowledge is especially beneficial in environments with a high degree of dynamism. In other studies the robustness of the policies concerning the quality of advance knowledge was investigated. Ferrucci et al. (2012) identified a near-linear relationship between the quality of the advance knowledge and the improvement in solution quality of a pro-active compared to a purely reactive approach. As a basis for their analysis, they proposed a structural diversity measure and liked the advantages of pro-active planning to the geographic and temporal variations of request arrivals for relocating the vehicles.

Hentenryck et al. (2010) investigated the effect of noisy stochastic information on the solution quality. A main finding was that it is better to rather over-estimate the number of appearing customers when it comes to optimizing the service quality. However, at some point the driven distance increases due to an increasing number of pro-active relocations where no customer materializes and a reactive planning outperforms the pro-active approach.

In this study, a dynamic pickup and delivery problem with stochastic customers is investigated in which the uncertainty related to the appearance of new customers changes over time. In particular, a standard pickup and delivery problem with time windows (PDPTW) is used as a problem formulation where requests appear dynamically during operations. Additionally, for each dynamically appearing customer an appearance probability between 0% and 100% is given according to a uniform distribution. The main objective is the minimization of the fleet size while the secondary objective is the minimization of the driven distance. Each used vehicles is weighted with a cost factor of $c_v = 3000$ while each unit of driven distance with a factor of $c_d = 1$. The objective value is thus a weighted sum of total fleet size and driven distance.

The proposed measure for data quality in this context is depending on the appearance probability over all stochastic customers. In practical contexts, advance information about future events might be noisy and the data quality can change over time. For example, long-term events might be associated with a larger uncertainty than near-term events. An important aspect of solution methods that are applied in uncertain environments is thus their robustness concerning data quality. The proposed methodology combines different pro-active policies to a portfolio and adaptively selects an appropriate one depending on the current data quality and instance characteristics. The performance of the presented methodology is evaluated on dynamic and stochastic PDPTW instances with different spatitial and temporal characteristics where the appearance probabilities of the requests change over time (see Section 6.6 for details on the problem instances).

As stated in the previous section, the problem characterization, the portfolio design, and the problem to algorithm mapping are important design decisions when specializing the general framework for portfolio-based algorithm selection to an application area. These aspects are detailed in the following in the context of this study.

5.2.2 Problem Characterization

The different problem instances are characterized according to three feature dimensions while all three are problem dependent. The first two dimensions

are temporal and spatial properties of the transport requests which are assumed to be fixed for each instance. The third dimension is the uncertainty in terms of request appearance which changes over time for a given instance.

As detailed in Section 6.6, the instances are split into pre-defined classes in terms of geographic distribution of requests (C = clustered, R = random, RC = mixed) and time window size (1 = tight, 2 = large). These characteristics are known a-priori per instance class and do not change. The combination of these characteristics leads to six different instance classes.

The uncertainty in terms of appearance of future requests changes over time which can be interpreted as a changing data quality of the a-priori information. For each request $r \in R_t$ in the future requests set, an appearance probability p_r is given as well as its arrival time a_r. The future request set R contains all stochastic requests at time t during the total planning horizon H.

A single uncertainty value is derived from the appearance probabilities of all future requests by aggregating them using three different formulas that weigh the requests uniformly, linear and exponentially respectively. The latter two consider the relative appearance time of request r defined as $\frac{a_r - t}{H}$. In that case, the uncertainty of near-term requests is considered more important than the uncertainty of long-term requests in calculating the total uncertainty value.

The uniform weighting function simply calculates the average over the appearance probabilities of all future requests neglecting their appearance time:

$$uncertaintyUniform(t) = 100\% - \sum_{r \in R_t} \frac{p_r}{|R_t|} \tag{5.1}$$

One way to consider the appearance time is to linearly decrease the weight of a future request with its distance to the current time t:

$$uncertaintyLinear(t) = 100\% - \frac{\sum_{r \in R_t} p_r (1.0 - \frac{a_r - t}{H})}{\sum_{r \in R_t} 1.0 - \frac{a_r - t}{H}} \tag{5.2}$$

Also an exponential decrease of weight is considered, while the parameter α determines the slope of the function:

$$uncertaintyExponential_{\alpha}(t) = 100\% - \frac{\sum_{r \in R_t} p_r e^{(-\frac{a_r - t}{H}/\alpha)}}{\sum_{r \in R_t} e^{(-\frac{a_r - t}{H}/\alpha}} \tag{5.3}$$

The different weighting functions are illustrated in Figure 5.4. The exponential weighting function with an α value of 0.1 provides the most short-term weighting while the uniform function weights all stochastic requests equally.

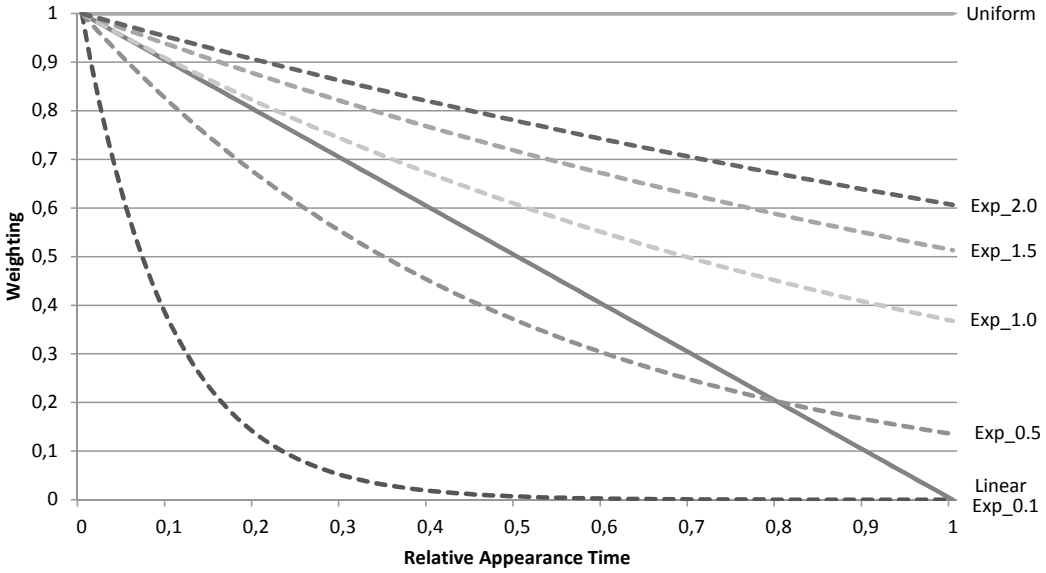

Figure 5.4: Different weighting functions to aggregate the appearance probabilities of all future requests to a single uncertainty value. In the linear and exponential (exp) weighting, the uncertainty of near-term requests is weighted more than long-term requests.

5.2.3 Portfolio Design

The portfolio design has been performed a-priori using a set of pre-defined heuristics that have different strengths and weaknesses concerning robustness behavior in terms of uncertainty. The algorithms have been evaluated beforehand on a set of evaluation instances with fixed characteristics to draw conclusions about their performance under different situations. A sampling-based approach, waiting strategies, and purely reactive planning have been combined to an algorithm portfolio to minimize the risk of providing bad solutions when the data quality decreases.

All three heuristics are based on a unified tabu search proposed by Cordeau and Laporte (2003) for route calculation which works on a standard pickup and delivery problem with time windows formulation. At each time step, the current situation is converted to a static problem. The initial solutions are created using a push-forward insertion heuristic proposed by Li and Lim (2001). The neighborhood is explored exhaustively and a maximum of 100 iterations are performed each time step. This basic algorithm is used as a purely re-active algorithm that can be extended with the consideration of a-priori knowledge about future requests.

Two different approaches, the multiple-scenario algorithm (MSA) and

waiting strategies, are integrated that utilize the available stochastic information about future requests differently. The MSA samples multiple scenarios from the stochastic data and solves each resulting deterministic problem instance separately. From the solutions of the individual scenarios a consensus plan is formed. This heuristic for stochastic vehicle routing problems has been proposed by Hentenryck and Bent (2009). The waiting strategies perform anticipatory waiting at strategically beneficial positions which are derived from the a-priori knowledge. The generated waiting strategies presented in Section 4.4 are used which have been specialized to the different spatial and temporal characteristics of the problem instance classes.

The three routing heuristics have been analyzed in terms of robustness behavior on different instance classes. For that purpose, evaluation instances with a fixed level of uncertainty were used. The evaluation set contains instances in six classes with different temporal and spatial characteristics. In each class, problems were considered where all customers have a certain appearance probability. Instances with 10% to 90% appearance probability were considered (details on the evaluation instance set can be found in Section 6.6).

The results on the evaluation set are illustrated in Figure 5.5 and illustrate a good potential for synergy in some cases. While for the clustered instances (C1, C2) the generated waiting strategies clearly dominate in nearly all instances but the ones with 90% appearance probability. For the other classes no single heuristic dominates in all cases. For example, for the class R1 the MSA algorithm dominates until about 30% appearance probability and then reactive planning yields better performance. The a-priori performance evaluation clearly indicates that a combination of different approaches might proof fruitful in terms of robustness.

5.2.4 Problem to Algorithm Mapping

From this a-priori performance analysis of the heuristics under different fixed problem characteristics, a fixed set of rules for algorithm selection can be derived beforehand. The rule set follows the logic of always selecting the heuristic that performed best on the evaluation instance set under certain characteristics.

The rule set is listed in Table 5.1 and has been derived manually from the performance evaluation that was performed during portfolio design. As a general pattern it can be observed, that the MSA heuristic performs best in all instance classes until a certain degree of uncertainty. Then, either the waiting strategies (AW) offer a larger robustness or it makes sense to switch to reactive planning (AH).

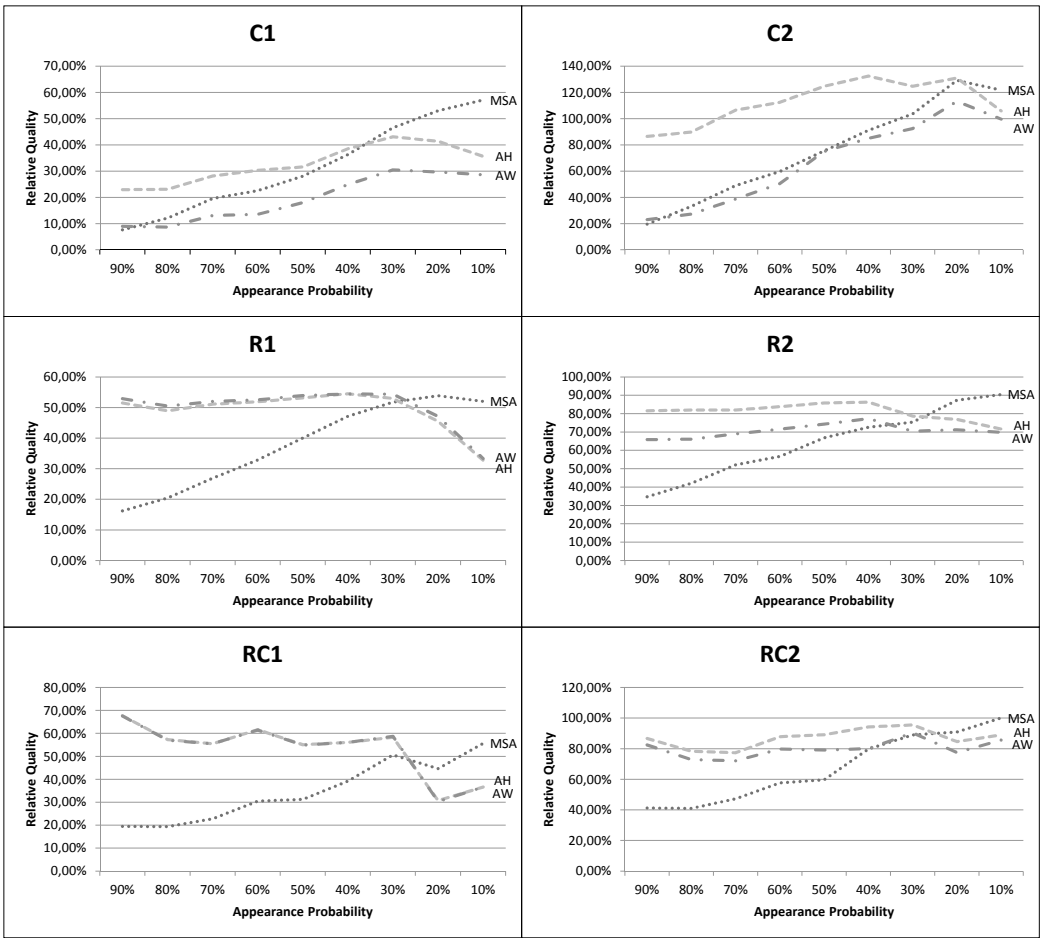

Figure 5.5: Performance evaluation of the three routing heuristics (reactive = AH, sampling = MSA, waiting = AW) on the evaluation instance set with fixed characteristics. Instance classes were considered with clustered (C_), randomly (R_), and mixed (RC_) geographical distributions as well as small (_1) and large (_2) time windows. In each of the six classes, instances with 90% to 10% appearance probability were examined where all customers have the same fixed uncertainty.

Class	Rule	Selection
C1	IF uncertainty <= 10%	MSA
	ELSE	AW
C2	IF uncertainty <= 10%	MSA
	ELSE	AW
R1	IF uncertainty <= 70%	MSA
	ELSE	AH
R2	IF uncertainty <= 60%	MSA
	ELSE	AW
RC1	IF uncertainty <= 70%	MSA
	ELSE	AW
RC2	IF uncertainty <= 70%	MSA
	ELSE	AW

Table 5.1: Rule set for the different instance classes for selecting a solution strategy depending on the current degree of uncertainty

5.2.5 Conclusions

Based on the results presented in Section 6.6, the overall conclusion can be drawn that the portfolio approach clearly outperforms any individual heuristic on the overall instance set. The expontential weighting function with an α value of 1.5 was identified as the best overall strategy. It can be observed that for small time windows a smaller α value of 0.5 is slightly advantageous because it focuses more on short term requests. In general, the α value of 1.5 proved as a robust parametrization and this portfolio achieved 2.16% better results than the MSA which was the best individual heuristic.

In terms of optimization potential it can be observed, that the individual policies complement each other especially in terms of spatial characteristics. While for the clustered problem instances the waiting strategies are clearly dominating the other strategies, the MSA is the overall best on the other classes until a certain degree of uncertainty. If the policy would be solely selected based on the instance class, selecting WS for clustered instances and MSA for the remaining while disregarding the current uncertainty, already a saving of 1.54% can be achieved compared to only using the MSA. If a dynamic policy selection is performed based on the current degree of uncertainty, this result can be further improved by 0.63% allowing the utilization of additional optimization potential.

However, the potential of the dynamic algorithm selection based on the changing degree of uncertainty clearly depends on the fact how well the performance profiles complement each other. In cases where one algorithm

dominates all others the potential savings are little. On the instances in the C1 and C2 classes, the waiting strategies (AW) perform basically equally well as the portfolio since it clearly outperforms all other algorithms under that conditions. For the other classes, however, no single policy dominates the others and it depends on the degree of uncertainty which algorithm should be chosen. In that case, it is advisable to combine the policies to a portfolio for dynamically choosing a policy based on the current data quality.

Summarizing, an important point is that the semi-automatically generated waiting policies complement the human-designed MSA policy. While the MSA proves as a generally robust policy, the waiting policies are highly specialized and prove beneficial in certain situations. A portfolio approach allows the utilization of additional optimization potential by the combination of specialized policies instead of focusing solely on overall performance.

Chapter 6

Computational Results

The computational experiments are carried out with the open-source optimization environment HeuristicLab in which the presented algorithmic concepts have been implemented. Its flexible architecture and design, as outlined by Wagner et al. (2014), are the foundation for the integration of the simulation optimization as well as the algorithmic approaches. The user interface offers capabilities for experiment design and results analysis. Parallel execution of the test runs is supported by the distributed computing infrastructure Hive (Scheibenpflug et al., 2012) integrated in HeuristicLab.

The test runs are executed on the Hive system running on a heterogeneous environment consisting of the HPC Blade Cluster as well as on lab computers provided by the University of Applied Sciences Upper Austria. The Blade cluster consists of 8 racks, each with two four-core Intel Xeon CPUs with 2.49 gigahertz accessing 32 gigabyte of memory and running a 64-bit Windows Server 2008 operating system. The lab computers are 22 desktop PCs, each with an four-core Intel Core2 CPU running at 2.66 gigahertz accessing 4 GB of memory and running a 64 bit Windows 7 Professional operating system.

The computational results are analyzed for statistical significance using the open-source software R (R Core Team, 2013). The analysis of the results is based on the findings of García et al. (2009) concerning the application of statistical techniques to evaluate the performance of metaheuristics. In particular, a multi-sample analysis is applied to compare the performance of several heuristics on different problem instances. For each instance, the results are aggregated to a single value (e.g., by building an arithmetic average over the runs for an instance). For comparing multiple algorithms, the Friedman test and for comparing two algorithms, the Wilcoxon signed rank test is applied on the median results of the test runs. In both cases, a significance level of 5% is used and for multiple comparisons the Hochberg procedure is applied to adjust the significance levels.

6.1 Identification of Bottlenecks Within Transport Activities in Steel Production

The methodology outlined in Section 3.2 was implemented in HeuristicLab to perform a simulation study with the aim of identifying bottlenecks within material handling of the cold charge steel production process. Both the simulation and optimization environment were modeled in HeuristicLab according to the generic pickup and delivery model presented in Section 3.1.2.

The data needed for the simulation model has been exported directly from the enterprise resource planning (ERP) system of the steel factory. Concretely, the crane movements at the handover places, the status of the storage places in the slab yard, the rolling schedule, the availability of the straddle carriers, and the processing schedule for the slabs has been exported. The simulation environment has been validated with domain experts of the steel factory aided by a visualization of the transport activities and different reporting capabilities.In total, data for nine different shifts has been exported and validated were each shift lasts eight hours.

The experiments have been carried out by individually simulating each shift. The original transport schedule was created by a human expert during the shift. On average, around 300 transport requests are carried out by 3 to 5 straddle carriers during a shift. The original schedule is a starting solution for the optimization algorithm which aims to improve it in terms of throughput while considering all operational constraints. Newly created schedules can be evaluated either by a full simulation run or by a static evaluation of the shift using the exported data. All experiments have been carried out on an Intel Xeon® Processor with 8 CPUs (2.5GHz) and 32 GB of memory.

The analysis of the original schedules executed during the nine shifts (shift A - I) are summarized in Table 6.1. The total effort for performing the transport activities during a shift is the sum of the travel and the service effort. On average, the straddle carriers spend about 57.21% of the total effort with traveling between the different locations and about 42,79% with service activities. The service activities include picking up slabs from stacks, shuffling activities to retrieve a slab that is not on top of a stack, and putting slabs on stacks. These results indicate that a significant time of the total effort is spent with service operations.

The results of the optimized schedule, that has been created by the algorithm for the different shifts are listed in Table 6.2. On average, the algorithm was able to save 4.48% of the total effort while 3.33% were saved on the travel and 6.01% were saved on the service effort. The travel effort in the optimized shifts is about 57,90% while the service effort is about 42.10%.

	Original		
	Travel Effort (minutes)	Service Effort (minutes)	Total Effort (minutes)
Shift A	474.16	556.5	1030.66
Shift B	774.9	431.6	1206.5
Shift C	678.05	563	1241.05
Shift D	616.02	260	876.02
Shift E	548.04	499	1047.04
Shift F	643.74	395.2	1038.94
Shift G	550.03	508	1058.03
Shift H	365.12	285	650.12
Shift I	532.15	378	910.15
Total	5182.21	3876.3	9058.51

Table 6.1: Travel and service effort of the original schedules created by the domain experts during different shifts.

	Optimized					
	Travel Effort (minutes)		Service Effort (minutes)		Total Effort (minutes)	
Shift A	459.32	(-3.13%)	507	(-8.89%)	966.32	(-6.24%)
Shift B	758.64	(-2.10%)	410.8	(-4.82%)	1169.44	(-3.07%)
Shift C	635.07	(-6.34%)	531	(-5.68%)	1166.07	(-6.04%)
Shift D	590.26	(-4.18%)	256	(-1.54%)	846.26	(-3.40%)
Shift E	523.27	(-4.52%)	483	(-3.21%)	1006.27	(-3.89%)
Shift F	635.12	(-1.34%)	361.4	(-8.55%)	996.52	(-4.08%)
Shift G	541.43	(-1.56%)	476	(-6.30%)	1017.43	(-3.84%)
Shift H	348.98	(-4.42%)	267	(-6.32%)	615.98	(-5.25%)
Shift I	517.69	(-2.72%)	351	(-7.14%)	868.69	(-4.56%)
Total	5009.78	(-3.33%)	3643.2	(-6.01%)	8652.98	(-4.48%)

Table 6.2: Comparison of the travel and service effort of the optimized with the original schedules.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**travel efforts** of the *original* ($\widetilde{x_1} = 550.03$) and the *optimized* ($\widetilde{x_2} = 541.43$) schedules	0.00195	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**service efforts** of the *original* ($\widetilde{x_1} = 431.6$) and the *optimized* ($\widetilde{x_2} = 410.8$) schedules	0.00195	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**total efforts** of the *original* ($\widetilde{x_1} = 1038.94$) and the *optimized* ($\widetilde{x_2} = 996.52$) schedules	0.00195	Reject H_0

Table 6.3: The efforts of the schedules have been compared using a one-sided Wilcoxon signed rank test and the results have been tested for statistical significance with $p < 0.05$.

Based on the statistical analysis presented in Table 6.3, it can be concluded that the optimized schedule significantly reduces both the travel and the service effort. In the following, a more detailed comparison will be performed to understand how the improvement is achieved to identify bottlenecks and optimization potential in terms of travel and service effort.

In Table 6.4, the travel effort between the original and the optimized schedules is examined in more detail. It can be observed, that in the optimized schedules the number of trips increases by 4.17%, however the time the straddle carries travel without having a slab loaded decreases by 10.68% on average. Based on the statistical tests presented in Table 6.5, it can be stated that the optimized schedule performs significantly more trips however reduces the time that the straddle carriers travel empty.

In terms of the service effort, especially operations at the slab yard are interesting since shuffling operations can occur there. In Table 6.6, the service effort for picking up and storing slabs at the yard is analyzed. The shuffling operations make up 10.81% while creating temporary stacks makes up 18.43% of the total service time. The rest of the service time is made up of single lifts while retrieving or delivering slabs to the yard or handover places. In the optimized schedule, a decrease of the shuffling effort (-5.44%) and effort in the creation of temporary stacks (-30.65%) can be observed. The results were tested for significance as reported in Table 6.7. While the decrease in shuffling effort is not significant, the decrease of stacking effort is.

	Original		Optimized			
	Number of Trips (count)	Traveled Empty (minutes)	Number of Trips (count)		Traveled Empty (minutes)	
Shift A	141	165.13	161	(+14.18%)	140.92	(-14.66%)
Shift B	210	291.79	221	(+5.24%)	266.80	(-8.56%)
Shift C	177	224.75	178	(+0.56%)	187.80	(-16.44%)
Shift D	170	177.41	172	(+1.18%)	164.10	(-7.50%)
Shift E	186	181.21	191	(+2.69%)	157.68	(-12.99%)
Shift F	186	252.20	189	(+1.61%)	241.11	(-4.40%)
Shift G	184	167.49	194	(+5.43%)	156.50	(-6.56%)
Shift H	96	98.28	102	(+6.25%)	77.51	(-21.14%)
Shift I	160	209.77	165	(+3.13%)	186.81	(-10.95%)
Total	1510	1768.04	1573	(+4.17%)	1579.23	(-10.68%)

Table 6.4: Analysis of the original and optimized schedules in terms of travel effort.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} > 0$	**number of trips** of the *original* ($\widetilde{x_1} = 177$) and the *optimized* ($\widetilde{x_2} = 178$) schedules	0.00195	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**empty minutes traveled** of the *original* ($\widetilde{x_1} = 181.21$) and the *optimized* ($\widetilde{x_2} = 164.1$) schedules	0.00195	Reject H_0

Table 6.5: By means of a one-sided Wilcoxon signed rank test, significance tests were performed for the travel efforts. A significance level of $p < 0.05$ was used.

	Original		**Optimized**			
	Shuffling Effort (minutes)	Temporary Stacking (minutes)	Shuffling Effort (minutes)		Temporary Stacking (minutes)	
Shift A	104	214.5	104	(+0.00%)	162	(-24.48%)
Shift B	28.6	39	28.6	(+0.00%)	15.6	(-60.00%)
Shift C	75	110	69	(-8.00%)	90	(-18.18%)
Shift D	42	80	42	(+0.00%)	77	(-3.75%)
Shift E	36	76	36	(+0.00%)	58.5	(-23.03%)
Shift F	28.6	46.8	20.8	(-27.27%)	27.3	(-41.67%)
Shift G	48	76	45	(-6.25%)	38	(-50.00%)
Shift H	33	36	27	(-18.18%)	18	(-50.00%)
Shift I	24	36	24	(+0.00%)	9	(-75.00%)
Total	419.2	714.3	396.4	(-5.44%)	495.4	(-30.65%)

Table 6.6: Analysis of the original and optimized schedules in terms of service effort at the slab yard.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**shuffling efforts** of the *original* ($\widetilde{x_1} = 36$) and the *optimized* ($\widetilde{x_2} = 36$) schedules	0.06250	Not reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**temporary stacking efforts** of the *original* ($\widetilde{x_1} = 76$) and the *optimized* ($\widetilde{x_2} = 38$) schedules	0.00195	Reject H_0

Table 6.7: With a significance value of $p < 0.05$, the statements about shuffling effort have been tested for statistical significance using a one-sided Wilcoxon signed rank test.

6.1.1 Scenario With Dynamically Arriving Information

	Optimized (with dynamic information)					
	Travel Effort (minutes)		Service Effort (minutes)		Total Effort (minutes)	
Shift A	467.73	(-1.36%)	537	(-3.50%)	1004.73	(-2.52%)
Shift B	789.66	(+1.90%)	434	(+0.56%)	1223.66	(+1.42%)
Shift C	663.71	(-2.11%)	537.5	(-4.53%)	1201.21	(-3.21%)
Shift D	612.49	(-0.57%)	253	(-2.69%)	865.49	(-1.20%)
Shift E	554.12	(+1.11%)	470	(-5.81%)	1024.12	(-2.19%)
Shift F	628.63	(-2.35%)	356.2	(-9.87%)	984.83	(-5.21%)
Shift G	553.17	(+0.57%)	488	(-3.94%)	1041.17	(-1.59%)
Shift H	362.86	(-0.62%)	277.5	(-2.63%)	640.36	(-1.50%)
Shift I	553.44	(+4.00%)	384	(+1.59%)	937.44	(+3.00%)
Total	5185.81	(+0.07%)	3737.2	(-3.59%)	8923.01	(-1.50%)

Table 6.8: Comparison of the travel and shuffling effort of the optimized schedule with the original schedules in the case of dynamically arriving information.

Another influence factor that has been analyzed in addition to the traveling and shuffling effort is the value of information. It is based on the assumption, that all transport requests apart from the rolling schedule are known one hour in advance in contrast to being known at the beginning of the shift as assumed in the previous section. When considering dynamically arriving information (see Table 6.2), the algorithmic performance decreases which is outlined in Table 6.8. The travel effort increases by 0.07% while the shuffling effort decreases by 3.59% compared to the original schedule. Compared to the optimized schedule without dynamic information, the travel effort increased by 3.51% while the shuffling effort increased by 2.58% leading to an increase of 3.12% in total effort.

The results have been tested for significance as listed in Table 6.9. No significant difference can be detected between the travel as well as the total efforts. A statistical significance exists between the service efforts compared to the original schedule.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**travel efforts** of the *original* ($\widetilde{x_1} = 550.03$) and the *dynamically optimized* ($\widetilde{x_2} = 554.12$) schedules	0.50000	Not reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**service efforts** of the *original* ($\widetilde{x_1} = 431.6$) and the *dynamically optimized* ($\widetilde{x_2} = 434$) schedules	0.00977	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**total efforts** of the *original* ($\widetilde{x_1} = 1038.94$) and the *dynamically optimized* ($\widetilde{x_2} = 1004.73$) schedules	0.10156	Not reject H_0

Table 6.9: The hypotheses about the efforts have been tested with a significance level of $p < 0.05$ by using the one-sided Wilcoxon signed rank test comparing the original with the dynamically optimized schedule

6.2 Analyzing the Influence Between Warehousing and In-House Transport in the Production of Firefighting Vehicles

To analyze the influences between warehousing and in-house transport, an integrated model was created as detailed in Section 3.3. It consists of the storage optimization algorithm implemented in HeuristicLab and the picking simulation implemented in AnyLogic©(Kofler et al., 2010). The release times obtained in the picking simulation are converted to daynamically appearing transport requests in the pickup and delivery simulation.

The test scenario was exported from the company enterprise resource planning (ERP) system and a production snapshot of a single day was created. During that day of operation, 487 parts are picked from the high-rack storage consisting of 12 two-sided aisles. They are commissioned in 89 pallets to the handover zone. These pallets are transported to 35 workstations which are served from the warehouse. Additionally, there are 89 uniformly distributed back-haul requests from the workstations to the warehouse leading to a total of 178 in-house transport requests. Based on this scenario, different storage assignment strategies as well as picking schedules have been evaluated.

In terms of storage assignment strategies, five random storage assignments have been generated (R1-R5) as well as assignments based solely on pick frequency (PF) and part affinity (PA). Also, combinations of PF and PA were tested weighting them differently ($quality = \alpha * PF + \beta * PA$). For the strategy C1 the parameters $\alpha = 1, \beta = 1$, for the strategy C2 $\alpha = 100, \beta = 1$, and for the strategy C3 $\alpha = 200, \beta = 1$ were used.

In terms of picking schedules, balanced and clustered picking schedules were evaluated. Using the balanced schedule, the orders are picked in a random sequence while using the clustered schedule orders are clustered according to their target workstation.

The results for the balanced schedule are listed in Table 6.10 while the results for the clustered schedule are listed in Table 6.11. Both the average picking duration in terms of warehousing and also the average transport duration in terms of in-house transport are analyzed.

The total quality can be evaluated by considering the total makespan of picking and transporting all items from and to the warehouse in the considered day of operation. Over all storage assignment strategies, the average picking duration increases by 9.79% while the average transport duration decreases by 8.75% leading to ta total reduction of makespan of 2.94% when using the clustered schedule compared to the balanced picking schedule.

	Balanced Picking Schedule		
	Average Picking Duration (minutes)	Average Transport Duration (minutes)	Total Makespan (minutes)
R1	4.71	10.25	501.00
R2	7.39	12.82	714.00
R3	3.91	9.81	479.00
R4	3.94	9.87	474.00
R5	4.82	10.42	494.00
PF	5.64	11.11	570.00
PA	5.64	10.78	560.00
C1	4.12	9.68	474.00
C2	4.47	10.44	494.00
C3	4.71	10.18	490.00
Median	4.71	10.34	494.00

Table 6.10: Evaluation of picking durations in the warehouse, in-house transport durations, and total makespan using a balanced picking schedule.

	Clustered Picking Schedule					
	Average Picking Duration (minutes)		Average Transport Duration (minutes)		Total Makespan (minutes)	
R1	5.51	(+16.95%)	9.51	(-7.23%)	506.00	(+1.00%)
R2	6.00	(-18.84%)	9.86	(-23.02%)	557.00	(-21.99%)
R3	4.44	(+13.51%)	8.95	(-8.79%)	473.00	(-1.25%)
R4	4.34	(+9.97%)	9.31	(-5.67%)	473.00	(-0.21%)
R5	5.25	(+8.86%)	9.43	(-9.50%)	483.00	(-2.23%)
PF	7.96	(+41.04%)	9.43	(-15.17%)	731.00	(+28.25%)
PA	6.71	(+18.92%)	10.40	(-3.53%)	626.00	(+11.79%)
C1	4.87	(+17.98%)	9.28	(-4.12%)	472.00	(-0.42%)
C2	4.25	(-5.03%)	9.53	(-8.70%)	473.00	(-4.25%)
C3	5.09	(+8.11%)	9.43	(-7.37%)	476.00	(-2.86%)
Median	5.17	(+9.79%)	9.43	(-8.75%)	479.50	(-2.94%)

Table 6.11: Comparison of warehouse picking, in-house transport, and total makespan of the clustered with the balanced picking schedule.

As listed in Table 6.12, statistical significances were detected in terms of an increase of picking and a decrease of transport durations when applying a clustered picking schedule. A significant decrease in total makespan can only be detected when the PF and PA scenarios are omitted which show an outlining increase in warehousing effort.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} > 0$	**picking durations** of the *balanced* ($\widetilde{x_1} = 4.71$) and the *clustered* ($\widetilde{x_2} = 5.17$) schedules	0.04199	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**transport durations** of the *balanced* ($\widetilde{x_1} = 10.34$) and the *clustered* ($\widetilde{x_2} = 9.43$) schedules	0.00098	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**total makespans** of the *balanced* ($\widetilde{x_1} = 494$) and the *clustered* ($\widetilde{x_2} = 479.5$) schedules	0.27830	Not reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} < 0$	**total makespans except PF, PA** of the *balanced* ($\widetilde{x_1} = 492$) and the *clustered* ($\widetilde{x_2} = 474.5$) schedules	0.01953	Reject H_0

Table 6.12: Significance of makespan difference between the balanced and the clustered schedules. The hypothesis have been tested using a one-sided Wilcoxon signed rank test with a significance level of $p < 0.05$.

6.3 Simulation-Based Sensitivity Analysis of Different Inventory Routing Scenarios

The simulation optimization approach detailed in Section 3.4 as well as the inventory replenishment policy generation detailed in Section 4.2 have been implemented in HeuristicLab to evaluate different inventory routing scenarios according to the generic inventory routing model presented in Section 3.1.2. Simulation optimization is mainly applied as a scenario technique in that context to perform a sensitivity analysis with respect to different endogenous as well as exogenous factors. Especially the consideration of mixed scenarios, where only part of the supermarkets have a vendor-managed inventory, is a novel aspect of this work.

The scenarios are evaluated in the context of a practically motivated case-study and have been created based on real-world data. The presented case-study considers the transportation of fast-moving consumer goods to supermarkets from a central depot and was conducted with an Austrian retailer who provided the data exported from the company enterprise resource planning system.

The scenario consists of 84 supermarkets which are served from a single central depot. The supermarkets are located in Lower and Upper Austria with a mean distance of 122 km between the customer locations (standard deviation 67 km) and a mean distance to the depot of 86 km (standard deviation 44 km). The supermarkets have been categorized into large, medium, and small according to the average weekly demand. In total, 5113 different products are considered while not necessarily each supermarkets offers all products. The supermarket categories are detailed in Table 6.13.

Category	Supermarkets	Products	Demand Fraction
Small	16	3559	11%
Medium	40	4272	44%
Large	28	4592	45%

Table 6.13: Supermarket categories in the considered case-study (cf. Vonolfen et al. (2013a)).

For each product, weekday, and customer category, separate distribution parameters (average, standard deviation) are given while the demand is assumed to follow a Gaussian distribution. No seasonal fluctuations are considered which means that the demand parameters do not change over time. A main challenge in the context is the fluctuation of the expected daily demand during a week which is illustrated in Figure 6.1. While on Wednesday and

Friday the highest demand occurs, on Sunday the supermarkets are closed.

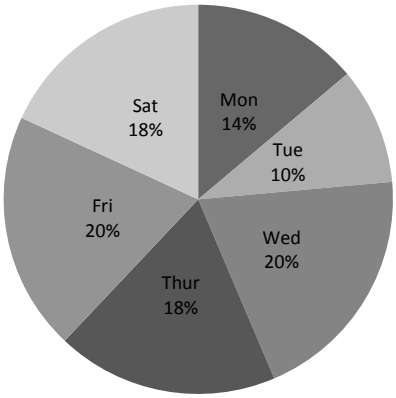

Figure 6.1: Fraction of the expected daily demand in terms of the expected weekly demand (cf. Vonolfen et al. (2013a).

The supermarkets are served using a fleet of homogeneous vehicles where each vehicle has a capacity of 30 roll containers. On Saturday no deliveries occur since the supermarkets are closed on Sunday. For each supermarket it can be chosen independently if a vendor-manged inventory should be used or the supermarket places the orders itself. This allows the consideration of mixed scenarios.

Based on this case-study, the influence of using a vendor-managed inventory, the influence of the service quality, and the influence of different exogenous factors on the distribution costs is investigated. This is achieved by considering different scenarios and evaluating the optimization potential using the proposed simulation optimization methodology. For each scenario where VMI is applied, a separate parametrization of the inventory replenishment policies has been generated as outlined in Section 4.2 on a set of training instances. The test runs have been carried out on ten instances that have been sampled beforehand from the scenario. Each instance covers the planning horizon of a fiscal quarter. The results are listed in detail for the test phase and the performance on the training instances is indicated as a comparison.

6.3.1 Influence of Vendor-Managed Inventory

To evaluate the potential savings when transitioning supermarkets to a vendor-managed inventory (VMI), three different scenarios were examined. In the *0% VMI* scenario the supermarkets place the orders themselves and VMI is

not used. The *50% VMI* scenario consists of 50% randomly selected supermarkets that have a VMI while the rest apply a classic order strategy. In the *100% VMI* scenario, all supermarkets have been transitioned to a VMI. In all scenarios a service quality (SQ) of 99% is ensured.

The results for the *0% VMI* scenario are listed in Table 6.14. On average a fleet size of 48.5 vehicles and a driven distance of 527665.56 is required to reach the desired service quality. In the *50%VMI* scenario, 7.38% of the total costs could be saved as listed in Table 6.15. While the driven distance is increased slightly by 1.85%, the fleet size can be reduced by 29.69% with comparable service quality. In the *100% VMI* scenario, 14.73% of the total costs could be saved compared to not applying VMI as detailed in Table 6.16. The driven distance is decreased by 6.15% and the fleet size by 35.46% on average with comparable service quality.

The results have been tested for statistical significance which is detailed in Table 6.17. It is shown, that applying VMI has a significant influence on the total costs. In a pairwise post-hoc analysis, the results for all scenarios are distinguishable from each other.

	0% VMI			
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)
Instance 1	526053.51	49	99.70%	1493107.02
Instance 2	528257.51	50	99.70%	1506515.02
Instance 3	530315.11	48	99.69%	1492630.22
Instance 4	526201.56	45	99.71%	1457403.11
Instance 5	529142.89	51	99.70%	1517285.78
Instance 6	524835.33	49	99.68%	1490670.67
Instance 7	525151.44	48	99.69%	1482302.89
Instance 8	528859.47	48	99.69%	1489718.93
Instance 9	530093.49	48	99.70%	1492186.98
Instance 10	527745.24	49	99.70%	1496490.49
Average	527665.56	48.50	99.70%	1491831.11
Stdev	1998.68	1.58	0.01%	15526.12

Table 6.14: Evaluation of scenario where all supermarkets follow an order-based strategy and no vendor managed inventory is used. The results presented on the test instances.

	50% VMI				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	535801.18	35	99.75%	1386602.36	(-7.13%)
Instance 2	543147.67	34	99.71%	1392295.33	(-7.58%)
Instance 3	540405.20	33	99.74%	1377810.40	(-7.69%)
Instance 4	535421.67	34	99.71%	1376843.33	(-5.53%)
Instance 5	531998.98	33	99.75%	1360997.96	(-10.30%)
Instance 6	537506.02	35	99.66%	1390012.04	(-6.75%)
Instance 7	535584.82	35	99.74%	1386169.64	(-6.49%)
Instance 8	535881.89	33	99.77%	1368763.78	(-8.12%)
Instance 9	540192.13	35	99.71%	1395384.27	(-6.49%)
Instance 10	538276.20	34	99.74%	1382552.40	(-7.61%)
Average	537421.58	34.10	99.73%	1381743.15	(-7.38%)
Stdev	3195.10	0.88	0.03%	10790.77	

Table 6.15: Evaluation of scenario where a randomly selected half of the supermarkets follow an order-based strategy and the other half has a vendor-managed inventory. The costs are compared to the 0% VMI scenario (Table 6.14). The results are presented on the test instances. The performance on the training instances was 1331476.54 (-3.64% compared to the test results).

	100% VMI				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	495597.11	31	99.76%	1270194.22	(-14.93%)
Instance 2	494161.67	31	99.72%	1267323.33	(-15.88%)
Instance 3	494369.11	31	99.76%	1267738.22	(-15.07%)
Instance 4	498315.00	31	99.75%	1275630.00	(-12.47%)
Instance 5	493132.20	32	99.76%	1274264.40	(-16.02%)
Instance 6	496249.42	32	99.75%	1280498.84	(-14.10%)
Instance 7	496965.98	32	99.73%	1281931.96	(-13.52%)
Instance 8	495431.53	31	99.65%	1269863.07	(-14.76%)
Instance 9	494665.71	31	99.72%	1268331.42	(-15.00%)
Instance 10	493276.31	31	99.70%	1265552.62	(-15.43%)
Average	495216.40	31.30	99.73%	1272132.81	(-14.73%)
Stdev	1638.50	0.48	0.03%	5694.93	

Table 6.16: Evaluation of scenario where all supermarkets have a vendor-managed inventory. The costs are compared to the 0% VMI scenario (Table 6.14). The results are presented on the test instances. The performance on the training instances was 1229812.78 (-3.33% compared to the test results).

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_1} = \widetilde{\mu_2} = \widetilde{\mu_3}$	$\neg H_0$	**Costs** of the scenarios: **0% VMI** ($\widetilde{x_1} = 1492408.60$) **50% VMI** ($\widetilde{x_2} = 1384361.02$) **100% VMI** ($\widetilde{x_3} = 1270028.65$)	4e-05	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} \neq 0$	Post-hoc pairwise comparisons:		
		0% / 50% VMI	0.002	Reject H_0
		0% / 100% VMI	0.002	Reject H_0
		50% / 100% VMI	0.002	Reject H_0

Table 6.17: Statistical analysis of the different VMI scenarios. A Friedman test was used to test if any median of the scenarios differ. For a post-hoc analysis, the Wilcoxon signed rank test was applied and the p-values were adjusted using Hochberg's procedure. A significance level of $p < 0.05$ was used.

6.3.2 Influence of Service Quality

While in the previous scenarios, a SQ of 99% was demanded, the influence of the SQ on the total costs was investigated by considering scenarios where only 95% and 90% SQ were demanded. In all three scenarios, 100% of the supermarkets apply VMI and thus the results are compared to the results with 100% VMI and 99% SQ as listed in Table 6.16.

The results for the scenario *95% SQ* are listed in Table 6.18. The average driven distance is reduced by 8.69% and the fleet size by 40.82%. However, the costs for the out-of stock situations are increased leading to total savings of 3.95% compared to the scenario with 99% SQ. In the scenario with *90% SQ*, as listed in Table 6.19, 11.08% are saved in terms of driven distance and 41.03% while the total costs are reduced by 6% compared to the scenario with 99% SQ.

The results have been tested for statistical significance as detailed in Table 6.20. The influence of the SQ on the resulting costs is significant and all pairwise compared scenarios lead to distinguishable results in terms of costs.

	95% Service Quality				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	478974.31	28	96.96%	1209948.62	(-4.74%)
Instance 2	479808.98	29	97.24%	1220617.96	(-3.69%)
Instance 3	483947.42	29	97.20%	1228894.84	(-3.06%)
Instance 4	482950.93	30	97.03%	1235901.87	(-3.11%)
Instance 5	479610.47	28	97.28%	1211220.93	(-4.95%)
Instance 6	486648.47	29	97.22%	1234296.93	(-3.61%)
Instance 7	480641.56	29	97.19%	1222283.11	(-4.65%)
Instance 8	482969.91	29	97.17%	1226939.82	(-3.38%)
Instance 9	481749.49	28	96.98%	1215498.98	(-4.17%)
Instance 10	480885.29	28	97.41%	1213770.58	(-4.09%)
Average	481818.68	28.70	97.17%	1221937.36	(-3.95%)
Stdev	2350.72	0.67	0.14%	9367.67	

Table 6.18: Scenario analysis with 95% service quality and 100% VMI supermarkets. The costs are compared with Table 6.16 where 99% service quality was demanded. The results are presented on the test instances. The performance on the training instances was 1189016.67 (-0.57% compared to the test results).

	90% Service Quality				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	468264.27	29	91.69%	1197528.53	(-5.72%)
Instance 2	469671.13	28	88.68%	1191342.27	(-6.00%)
Instance 3	468102.56	28	90.41%	1188205.11	(-6.27%)
Instance 4	466409.89	28	91.07%	1184819.78	(-7.12%)
Instance 5	468997.24	29	90.87%	1198994.49	(-5.91%)
Instance 6	466409.89	28	91.07%	1184819.78	(-7.47%)
Instance 7	468997.24	29	90.87%	1198994.49	(-6.47%)
Instance 8	471051.49	29	91.44%	1203102.98	(-5.26%)
Instance 9	469764.38	29	90.48%	1200528.76	(-5.35%)
Instance 10	474206.96	29	92.08%	1209413.91	(-4.44%)
Average	469187.50	28.60	90.87%	1195775.01	(-6.00%)
Stdev	2279.58	0.52	0.92%	8186.09	

Table 6.19: Scenario analysis with 90% service quality and 100% VMI supermarkets. The costs are compared with Table 6.16 where 99% service quality was demanded. The results are presented on the test instances. The performance on the training instances was 1210919.99 (-0.90% compared to the test results).

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_1} = \widetilde{\mu_2} = \widetilde{\mu_3}$	$\neg H_0$	**Costs** of the scenarios: **99% SQ** ($\widetilde{x_1} = 1270028.65$) **95% SQ** ($\widetilde{x_2} = 1221450.54$) **90% SQ** ($\widetilde{x_3} = 1198261.51$)	4e-05	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} \neq 0$	Post-hoc pairwise comparisons:		
		99% SQ / 95% SQ	0.002	Reject H_0
		99% SQ / 90% SQ	0.002	Reject H_0
		95% SQ / 90% SQ	0.002	Reject H_0

Table 6.20: Significance test for the scenarios with different service qualities. The differences between the samples was tested using a Friedman test. As a post-hoc procedure, a pair-wise Wilcoxon signed rank test was applied and the p-values were adjusted using Hochberg's procedure. For both tests, a significance level of $p < 0.05$ was used.

6.3.3 Influence of Exogenous Characteristics

While the service quality is and important endogenous factor, spatial and temporal scenario characteristics can be considered as exogenous factors. Concretely, the influence of the demand distribution in terms of number of products and heterogeneity of the demand and spatial distribution of the supermarkets are considered.

To investigate the influence of the different exogenous scenario properties, mixed scenarios are investigated where 50% of the supermarkets are transitioned to VMI while the remaining retain an order-based strategy. In the case of the *Most Products* scenario, the supermarkets with the largest product portfolio while in the *Demand Heterogeneity* scenario the supermarkets with the highest fluctuation in demand are selected. In terms of spatial influence factors, in the *Multiple Clusters* scenario five clusters of VMI supermarkets are created while in the *Single Cluster* scenario a single cluster is created according to geographic proximity. The single cluster / five clusters with the highest total demand are transitioned to VMI.

The results are compared to a random selection of the VMI supermarkets as listed in Table 6.15. The best result was achieved by selecting the largest supermarkets and, as listed in Table 6.21, 6.28% of the costs can be saved compared to random selection. Selecting multiple clusters (Table 6.23) and a single cluster (Table 6.24) leads to slight benefits while a single cluster is better than multiple clusters. Selecting VMI supermarkets in terms of demand heterogeneity is not beneficial in that case and even a random selection is better (Table 6.22). This can be explained by the fact that the heterogeneity of the smallest supermarkets is the highest in the considered case-study and thus the smallest supermarkets are selected in that case.

The statistical significance of the influence of the exogenous properties on the costs has been verified as listed in Table 6.25. By pairwise comparision, all scenarios but the *Random* and the *Multiple Clusters* scenario can be distinguished in terms of costs.

	Most Products				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	502923.20	32	99.39%	1293846.40	(-6.69%)
Instance 2	501022.27	32	99.39%	1290044.53	(-7.34%)
Instance 3	501713.58	33	99.27%	1300427.16	(-5.62%)
Instance 4	502945.98	32	99.47%	1293891.96	(-6.02%)
Instance 5	500362.20	33	99.05%	1297724.40	(-4.65%)
Instance 6	502708.62	32	99.33%	1293417.24	(-6.95%)
Instance 7	505101.67	32	99.17%	1298203.33	(-6.35%)
Instance 8	501554.51	32	99.59%	1291109.02	(-5.67%)
Instance 9	500620.44	32	99.36%	1289240.89	(-7.61%)
Instance 10	502309.24	33	99.50%	1301618.49	(-5.85%)
Average	502126.17	32.30	99.35%	1294952.34	(-6.28%)
Stdev	1399.53	0.48	0.16%	4331.37	

Table 6.21: Scenario with 50% VMI supermarkets, where the supermarkets with the most products where selected. The costs are compared to the results presented in Table 6.15 where the VMI supermarkets were randomly chosen. The results are presented on the test instances. The performance on the training instances was 1275082.02 (-1.53% compared to the test results).

	Demand Heterogeneity				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	544684.67	34	99.77%	1395369.33	(+0.63%)
Instance 2	543361.00	35	99.77%	1401722.00	(+0.68%)
Instance 3	551475.27	34	99.77%	1408950.53	(+2.26%)
Instance 4	546647.73	34	99.78%	1399295.47	(+1.63%)
Instance 5	542532.60	33	99.77%	1382065.20	(+1.55%)
Instance 6	545887.47	35	99.78%	1406774.93	(+1.21%)
Instance 7	543828.00	35	99.75%	1402656.00	(+1.19%)
Instance 8	546704.51	33	99.78%	1390409.02	(+1.58%)
Instance 9	543541.58	34	99.78%	1393083.16	(-0.16%)
Instance 10	544227.89	35	99.76%	1403455.78	(+1.51%)
Average	545289.07	34.20	99.77%	1398378.14	(+1.20%)
Stdev	2592.12	0.79	0.01%	8199.56	

Table 6.22: Scenario with 50% VMI supermarkets, where the supermarkets with the highest demand heterogeneity where selected. The costs are compared to the results presented in Table 6.15 where the VMI supermarkets were randomly chosen. The results are presented on the test instances. The performance on the training instances was 1349603.45 (-3.49% compared to the test results).

	Multiple Clusters				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	533721.51	33	99.76%	1364443.02	(-1.60%)
Instance 2	538095.20	35	99.64%	1391190.40	(-0.08%)
Instance 3	532705.67	33	99.76%	1362411.33	(-1.12%)
Instance 4	538613.56	33	99.78%	1374227.11	(-0.19%)
Instance 5	537107.42	34	99.77%	1380214.84	(+1.41%)
Instance 6	537703.00	34	99.72%	1381406.00	(-0.62%)
Instance 7	531739.91	34	99.79%	1369479.82	(-1.20%)
Instance 8	536026.93	36	99.76%	1396053.87	(+1.99%)
Instance 9	535661.42	35	99.76%	1386322.84	(-0.65%)
Instance 10	537158.82	35	99.77%	1389317.64	(0.49%)
Average	535853.34	34.20	99.75%	1379506.69	(-0.16%)
Stdev	2375.40	1.03	0.04%	11569.69	

Table 6.23: Scenario with 50% VMI supermarkets, where supermarkets from multiple geographical clusters were selected. The costs are compared to the results presented in Table 6.15 where the VMI supermarkets were randomly chosen. The results are presented on the test instances. The performance on the training instances was 1374399.78 (-0.37% compared to the test results).

	Single Cluster				
	Distance (km)	Fleet (count)	SQ (relative)	Costs (Euro)	
Instance 1	526859.87	34	99.75%	1359719.73	(-1.94%)
Instance 2	528458.73	34	99.65%	1362917.47	(-2.11%)
Instance 3	529617.98	33	99.50%	1356235.96	(-1.57%)
Instance 4	527769.60	34	99.76%	1361539.20	(-1.11%)
Instance 5	526658.16	35	99.61%	1368316.31	(+0.54%)
Instance 6	523386.71	33	99.75%	1343773.42	(-3.33%)
Instance 7	532431.82	33	99.71%	1361863.64	(-1.75%)
Instance 8	526266.00	34	99.75%	1358532.00	(-0.75%)
Instance 9	528929.98	34	99.70%	1363859.96	(-2.26%)
Instance 10	529556.29	34	99.54%	1365112.58	(-1.26%)
Average	527993.51	33.80	99.67%	1360187.03	(-1.56%)
Stdev	2431.24	0.63	0.09%	6699.14	

Table 6.24: Scenario with 50% VMI supermarkets, where supermarkets from a single geographical cluster were selected. The costs are compared to the results presented in Table 6.15 where the VMI supermarkets were randomly chosen. The results are presented on the test instances. The performance on the training instances was 1355699.13 (-0.33% compared to the test results).

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_1} = \widetilde{\mu_2} = \widetilde{\mu_3} = \widetilde{\mu_4} = \widetilde{\mu_5}$	$\neg H_0$	**Costs** of the scenarios: **Random** ($\widetilde{x_1} = 1384361.02$) **Most Products** ($\widetilde{x_2} = 1293869.18$) **Demand Het.** ($\widetilde{x_3} = 1400508.74$) **Mult. Clusters** ($\widetilde{x_4} = 1380810.42$) **Single Cluster** ($\widetilde{x_5} = 1361701.42$)	5e-07	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} \neq 0$	Post-hoc pairwise comparisons:		
		Random / Products	0.0098	Reject H_0
		Random / Demand	0.0117	Reject H_0
		Random / Clusters	0.5566	Not Reject H_0
		Random / Cluster	0.0117	Reject H_0
		Products / Demand	0.0098	Reject H_0
		Products / Clusters	0.0098	Reject H_0
		Products / Cluster	0.0098	Reject H_0
		Demand / Clusters	0.0117	Reject H_0
		Demand / Cluster	0.0098	Reject H_0
		Clusters / Cluster	0.0098	Reject H_0

Table 6.25: Statistical analysis of scenarios with 50% VMI supermarkets that were chosen according to different criteria. The influence of this factor was tested using a Friedman test and as a post-hoc procedure, a pair-wise Wilcoxon signed rank test was applied and the p-values were adjusted using Hochberg's procedure. A significance value of $p < 0.05$ is used.

6.4 Quality of Service and Runtime Analysis of Generated Priority Rules for Taxi Buses

The methodology described in Section 4.3 has been implemented in HeuristicLab using the simulation optimization model for the pickup and delivery problem described in Section 3.1.2 as well as the genetic programming implementation presented by Kommenda et al. (2012).

The methodology is evaluated on two different scenarios. Both consist of 26 predefined service points which are geographically uniformly distributed. There is a fleet of 6 taxi buses which can carry up to 10 people. In total, 1000 requests arrive during the planning horizon according to a Poisson process and each request has to be serviced within 15 minutes after it appeared. For the low intensity scenario 4 requests, while for the high intensity scenario 6 requests arrive per minute on average. For each scenario, 14 instances have been generated leading to a training set of 7 and a test set of 7 instances.

Using that test environment, five different dispatching policies have been evaluated. Three simple dispatching policies, as described in Section 2.2.1 have been are considered. The simple policies are the first come First come first serve (FCFS) policy that serves the requests according to their appearance, the earliest due date (EDD) policy that serves them according to their urgency, and the nearest neighbor (NN) policy which serves the request that is currently closest to the vehicle.

These strategies are compared to the complex policies proposed in Section 4.3, which are based on a linear and a tree representation, on the test set. Additionally, for the generated policies the results of the training phase are indicated as a comparison. Two planning-based algorithms serve as a bound for the policies. The a-priori tabu search solves a static problem instance where the requests do not appear dynamically but are known in advance. The ad-hoc tabu search is an online algorithm which recalculates the current set of routes every time a new request appears.

6.4.1 Low Intensity Scenario

The low-intensity scenario is characterized with 4 requests appearing per minute on average. The results for the a-priori tabu search presented in Table 6.26 serve as a competitive comparison to the dynamic approaches. The ad-hoc tabu search, which is an online algorithm that reoptimizes the current set of routes whenever new requests appear, performs on average 34.34% worse than the static algorithm (the results are listed in Table 6.27).

	Tabu Search (a-priori)			
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)	Runtime (minutes)
Instance 1 (static)	3361.58	0.22	3361.80	49.14
Instance 2 (static)	3518.22	0.75	3518.97	47.86
Instance 3 (static)	3457.37	0.22	3457.59	47.63
Instance 4 (static)	3354.71	0.54	3355.25	44.41
Instance 5 (static)	3269.69	1.46	3271.15	45.79
Instance 6 (static)	3389.95	0.62	3390.57	48.47
Instance 7 (static)	3228.12	0.00	3228.12	45.30
Average	3368.52	0.54	3369.06	46.94
Stdev	100.40	0.48	100.39	1.77

Table 6.26: Lead time and tardiness results for the ad-hoc tabu search on the low intensity test instances with all requests known in advance.

	Tabu Search (ad-hoc)				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	4466.83	0.00	4466.83	(+32.87%)	52.45
Instance 2	4703.26	0.00	4703.26	(+33.65%)	49.94
Instance 3	4650.33	0.80	4651.13	(+34.52%)	38.62
Instance 4	4465.75	0.14	4465.89	(+33.10%)	57.87
Instance 5	4416.00	0.00	4416.00	(+35.00%)	51.34
Instance 6	4532.38	0.12	4532.50	(+33.68%)	60.20
Instance 7	4462.25	0.00	4462.25	(+38.23%)	59.17
Average	4528.11	0.15	4528.27	(+34.41%)	52.80
Stdev	108.15	0.29	108.29		7.44

Table 6.27: Lead time and tardiness results for the ad-hoc tabu search on the low intensity test instances. The quality is compared to the a-priori tabu search (Table 6.26).

The results of the ad-hoc tabu search are used to compare this planning approach to different dispatching policies. In general, the evolved tree policy yields the best performance of all policies as listed in Table 6.32 and achieved 3% worse results than the online planning algorithm while using around 74 times less runtime. The tree dispatching policy achieved an increase in solution quality of 30.11% compared to the FCFS policy (Table 6.28), 24,52% compared to the EDD policy (Table 6.29), 99.59% compared to the NN policy (Table 6.30), and 1.68% compared to the evolved linear policy (Table 6.31).

The results presented in Table 6.33 illustrate, that there is a significant effect on the used policy on the solution quality. In a post-hoc pairwise comparison, the results for all policies could be distinguished from each other with a significance level of $p < 0.05$ and the evolved tree policy was confirmed as the best policy.

	FCFS				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	6224.71	11.08	6235.79	(+39.60%)	0.54
Instance 2	8830.73	1800.00	10630.73	(+126.03%)	0.67
Instance 3	5910.12	59.08	5969.20	(+28.34%)	0.54
Instance 4	5640.24	1.76	5642.00	(+26.34%)	0.62
Instance 5	5609.90	802.99	6412.89	(+45.22%)	1.07
Instance 6	6380.07	3.45	6383.52	(+40.84%)	0.50
Instance 7	5437.94	0.00	5437.94	(+21.87%)	0.58
Average	6290.53	382.62	6673.15	(+47.37%)	0.65
Stdev	1170.66	690.85	1783.72		0.20

Table 6.28: Lead time and tardiness results for the first come first serve policy on the low intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.27).

	EDD				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	6069.12	12.15	6081.27	(+36.14%)	0.66
Instance 2	8585.18	150.11	8735.29	(+85.73%)	0.66
Instance 3	5784.00	91.30	5875.30	(+26.32%)	0.56
Instance 4	5609.43	4.35	5613.78	(+25.70%)	0.53
Instance 5	5416.44	46.29	5462.73	(+23.70%)	0.97
Instance 6	6250.18	5.55	6255.73	(+38.02%)	0.64
Instance 7	5231.67	0.00	5231.67	(+17.24%)	0.53
Average	6135.15	44.25	6179.40	(+36.46%)	0.65
Stdev	1136.72	57.01	1181.26		0.15

Table 6.29: Lead time and tardiness results for the earliest due date policy on the low intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.27).

	NN				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	5031.56	1.1E+6	1.1E+6	(+2.4E+4%)	0.65
Instance 2	5301.24	5.9E+6	5.9E+6	(+1.3E+5%)	0.72
Instance 3	5046.62	7.5E+5	7.5E+5	(+1.6E+4%)	0.55
Instance 4	4778.20	7.8E+1	4.9E+3	(+8.7E+0%)	0.81
Instance 5	4778.32	1.1E+5	1.2E+5	(+2.5E+3%)	0.96
Instance 6	4999.61	1.1E+5	1.1E+5	(+2.4E+3%)	0.52
Instance 7	4898.36	4.8E+4	5.3E+4	(+1.1E+3%)	0.66
Average	4976.27	1.1E+6	1.1E+6	(+2.5E+4%)	0.70
Stdev	181.95	2.2E+6	2.2E+6		0.15

Table 6.30: Lead time and tardiness results for the nearest neighbor policy on the low intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.27).

	Linear				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	4678.16	0.00	4678.16	(+4.73%)	0.69
Instance 2	5014.52	0.00	5014.52	(+6.62%)	0.77
Instance 3	4805.72	36.29	4842.01	(+4.10%)	0.55
Instance 4	4601.19	0.00	4601.19	(+3.03%)	0.65
Instance 5	4624.89	11.20	4636.09	(+4.98%)	1.06
Instance 6	4742.46	16.46	4758.92	(+5.00%)	0.51
Instance 7	4675.46	0.21	4675.67	(+4.78%)	0.69
Average	4734.63	9.17	4743.79	(+4.76%)	0.70
Stdev	141.42	13.70	143.79		0.18

Table 6.31: Lead time and tardiness results for the evolved linear policy on the low intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.27). On the training instances, a gap of 4.26% to the planning algorithm was achieved.

	Tree				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	4666.94	0.00	4666.94	(+4.48%)	0.67
Instance 2	4722.24	19.16	4741.40	(+0.81%)	0.77
Instance 3	4724.18	18.56	4742.74	(+1.97%)	0.59
Instance 4	4561.06	0.00	4561.06	(+2.13%)	0.79
Instance 5	4535.26	21.24	4556.50	(+3.18%)	0.97
Instance 6	4734.31	15.42	4749.73	(+4.79%)	0.52
Instance 7	4629.63	0.00	4629.63	(+3.75%)	0.69
Average	4653.37	10.63	4664.00	(+3.00%)	0.71
Stdev	81.17	10.08	84.54		0.15

Table 6.32: Lead time and tardiness results for the evolved tree policy on the low intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.27). On the training instances, a gap of 2.24% to the planning algorithm was achieved.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_1} = \widetilde{\mu_2} = \widetilde{\mu_3} = \widetilde{\mu_4} = \widetilde{\mu_5}$	$\neg H_0$	**Quality** of the policies: **FCFS** $(\widetilde{x_1} = 6235.79)$ **EDD** $(\widetilde{x_2} = 5875.30)$ **NN** $(\widetilde{x_3} = 1.20E + 05)$ **Linear** $(\widetilde{x_4} = 4678.16)$ **Tree** $(\widetilde{x_5} = 4666.94)$	3.25e-05	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} \neq 0$	Post-hoc pairwise comparisons:		
		FCFS / EDD	0.031	Reject H_0
		FCFS / NN	0.031	Reject H_0
		FCFS / Linear	0.031	Reject H_0
		FCFS / Tree	0.031	Reject H_0
		EDD / NN	0.031	Reject H_0
		EDD / Linear	0.031	Reject H_0
		EDD / Tree	0.031	Reject H_0
		NN / Linear	0.031	Reject H_0
		EDD / Tree	0.031	Reject H_0
		Linear / Tree	0.031	Reject H_0

Table 6.33: Statistical analysis of the different policies for the low intensity scenario. A Friedman test was used to test if any median of the qualities achieved by the policies differ. For a post-hoc analysis, the Wilcoxon signed rank test was applied and the p-values were adjusted using Hochberg's procedure. A significance level of $p < 0.05$ was used.

6.4.2 High Intensity Scenario

In the high intensity scenario, 6 requests appear on average per minute as compared to 4 requests in the low intensity scenario. Since the maximum time for each request to be served is 15 minutes, the urgency increases. The online tabu search algorithm (Table 6.35) performs 14.97% worse than the static algorithm (Table 6.34). This indicates, that the optimization potential decreases compared to the low intensity scenario where the competitive analysis of the online algorithm resulted in a gap of 34.41%.

	Tabu Search (a priori)			
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)	Runtime (minutes)
Instance 1	5364.30	0.04	5364.34	145.18
Instance 2	5444.51	1.90	5446.41	167.05
Instance 3	5257.40	10.85	5268.25	108.00
Instance 4	4772.34	7.11	4779.45	76.46
Instance 5	4661.58	0.19	4661.77	73.42
Instance 6	5270.83	3.16	5273.99	123.08
Instance 7	4748.94	0.00	4748.94	76.07
Average	5074.27	3.32	5077.59	109.89
Stdev	331.83	4.18	332.41	37.15

Table 6.34: Lead time and tardiness results for the ad-hoc tabu search on the high intensity test instances with all requests known in advance.

Comparing the planning tabu search algorithm to the priority policies does not produce such a clear picture for the high intensity as in the low intensity scenarios. Both the evolved linear and tree policies show comparable performance as the planning algorithm. While the linear policy (Table 6.39) produced 0.69% worse performance, the tree policy (Table 6.40) achieved 0.34% better performance compared to the planning algorithm. The simple policies were not able to achieve feasible solutions and had a high tardiness penalty resulting from the high urgency of the requests. The evolved tree policy achieved 99.9992% better solution quality than the the FCFS policy (Table 6.36), 99.9993% than the EDD policy (Table 6.37), and 99.9111% than the NN policy (Table 6.38).

The statistical analysis comparing the policies for the high intensity scenario is listed in Table 6.41. The effect on the used policy on the achieved solution quality could be confirmed. The results of all policies can be distinguished in a pairwise comparison with a significance level of $p < 0.05$ except the FCFS / EDD policies as well as the Linear / Tree policies.

	Tabu Search (ad hoc)				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	6077.08	11.16	6088.24	(+13.49%)	56.42
Instance 2	6269.93	6.14	6276.07	(+15.23%)	52.59
Instance 3	5958.74	0.04	5958.78	(+13.11%)	58.88
Instance 4	5623.37	1.20	5624.57	(+17.68%)	49.82
Instance 5	5442.03	0.00	5442.03	(+16.74%)	56.84
Instance 6	5917.73	0.13	5917.86	(+12.21%)	60.86
Instance 7	5554.13	0.73	5554.86	(+16.97%)	98.98
Average	5834.72	2.77	5837.49	(+14.97%)	62.06
Stdev	302.22	4.29	304.96		16.70

Table 6.35: Lead time and tardiness results for the ad-hoc tabu search on the high intensity test instances. The quality is compared to the a-priori tabu search (Table 6.34).

	FCFS				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	68496.73	8.8E+8	8.8E+8	(+1.4E+7%)	0.98
Instance 2	62283.36	7.2E+8	7.2E+8	(+1.2E+7%)	1.11
Instance 3	80463.06	8.4E+8	8.4E+8	(+1.4E+7%)	1.06
Instance 4	37968.70	5.5E+8	5.5E+8	(+9.8E+6%)	0.99
Instance 5	37188.25	6.6E+8	6.6E+8	(+1.2E+7%)	0.81
Instance 6	77682.23	8.7E+8	8.7E+8	(+1.5E+7%)	0.93
Instance 7	42646.58	8.1E+8	8.1E+8	(+1.5E+7%)	0.60
Average	58104.13	7.6E+8	7.6E+8	(+1.3E+7%)	0.92
Stdev	18666.11	1.2E+8	1.2E+8		0.17

Table 6.36: Lead time and tardiness results for the first come first serve policy on the high intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.35).

	EDD				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	67929.05	8.8E+8	8.8E+8	(+1.4E+7%)	0.95
Instance 2	65949.72	7.6E+8	7.6E+8	(+1.2E+7%)	1.12
Instance 3	77237.39	8.5E+8	8.5E+8	(+1.4E+7%)	0.85
Instance 4	39189.79	5.8E+8	5.8E+8	(+1.0E+7%)	0.60
Instance 5	38121.74	7.4E+8	7.4E+8	(+1.4E+7%)	0.57
Instance 6	76331.70	8.7E+8	8.7E+8	(+1.5E+7%)	0.63
Instance 7	42173.65	8.0E+8	8.0E+8	(+1.4E+7%)	0.56
Average	58133.29	7.8E+8	7.8E+8	(+1.3E+7%)	0.76
Stdev	17641.01	1.0E+8	1.0E+8		0.22

Table 6.37: Lead time and tardiness results for the earliest due date policy on the high intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.35).

	NN				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	6959.14	8.2E+6	8.2E+6	(+1.3E+5%)	0.60
Instance 2	7349.36	9.1E+6	9.1E+6	(+1.4E+5%)	0.91
Instance 3	7303.37	6.1E+6	6.1E+6	(+1.0E+5%)	0.86
Instance 4	6886.02	6.4E+6	6.4E+6	(+1.1E+5%)	0.88
Instance 5	6216.72	1.5E+5	1.6E+5	(+2.7E+3%)	0.94
Instance 6	7188.67	1.1E+7	1.1E+7	(+1.9E+5%)	0.77
Instance 7	6623.19	4.7E+6	4.7E+6	(+8.4E+4%)	1.14
Average	6932.35	6.5E+6	6.5E+6	(+1.1E+5%)	0.87
Stdev	405.76	3.5E+6	3.5E+6		0.17

Table 6.38: Lead time and tardiness results for the nearest neighbor policy on the high intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.35).

	Linear				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	5896.75	0.00	5896.75	(-3.15%)	0.85
Instance 2	6398.23	34.85	6433.08	(+2.50%)	0.66
Instance 3	5968.89	0.00	5968.89	(+0.17%)	0.61
Instance 4	5667.32	0.02	5667.34	(+0.76%)	1.06
Instance 5	5551.68	0.00	5551.68	(+2.01%)	0.94
Instance 6	5822.69	0.00	5822.69	(-1.61%)	0.72
Instance 7	5744.79	57.56	5802.35	(+4.46%)	0.95
Average	5864.34	13.20	5877.54	(+0.69%)	0.83
Stdev	273.75	23.48	281.77		0.17

Table 6.39: Lead time and tardiness results for the evolved linear policy on the high intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.35). On the training instances, a gap of 1.60% to the planning algorithm was achieved.

	Tree				
	Lead Time (minutes)	Tardiness (penalty)	Quality (total)		Runtime (minutes)
Instance 1	5988.15	0.15	5988.30	(-1.64%)	0.47
Instance 2	6191.53	0.16	6191.69	(-1.34%)	0.47
Instance 3	5878.62	0.00	5878.62	(-1.35%)	0.50
Instance 4	5588.90	3.91	5592.81	(-0.56%)	0.58
Instance 5	5384.35	0.00	5384.35	(-1.06%)	0.53
Instance 6	5956.87	0.00	5956.87	(+0.66%)	0.48
Instance 7	5732.33	0.00	5732.33	(+3.19%)	0.53
Average	5817.25	0.60	5817.85	(-0.34%)	0.51
Stdev	270.59	1.46	270.10		0.04

Table 6.40: Lead time and tardiness results for the evolved tree policy on the high intensity test instances. The quality is compared to the ad-hoc tabu search (Table 6.35). On the training instances, a gap of -0.27% to the planning algorithm was achieved.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_1} = \widetilde{\mu_2} = \widetilde{\mu_3} = \widetilde{\mu_4} = \widetilde{\mu_5}$	$\neg H_0$	**Quality** of the policies: **FCFS** $(\widetilde{x_1} = 8.1E+08)$ **EDD** $(\widetilde{x_2} = 8.0E+08)$ **NN** $(\widetilde{x_3} = 6.4E+06)$ **Linear** $(\widetilde{x_4} = 5822.69)$ **Tree** $(\widetilde{x_5} = 5878.62)$	4.017e-05	Reject H_0
$\widetilde{\mu_D} = 0$	$\widetilde{\mu_D} \neq 0$	Post-hoc pairwise comparisons:		
		FCFS / EDD	0.469	Not Reject H_0
		FCFS / NN	0.047	Reject H_0
		FCFS / Linear	0.047	Reject H_0
		FCFS / Tree	0.047	Reject H_0
		EDD / NN	0.047	Reject H_0
		EDD / Linear	0.047	Reject H_0
		EDD / Tree	0.047	Reject H_0
		NN / Linear	0.047	Reject H_0
		EDD / Tree	0.047	Reject H_0
		Linear / Tree	0.469	Not Reject H_0

Table 6.41: Statistical analysis of the different policies for the high intensity scenario. A Friedman test was used to test if any median of the qualities achieved by the policies differ. For a post-hoc analysis the Wilcoxon signed rank test was applied and the p-values were adjusted using Hochberg's procedure. A significance level of $p < 0.05$ was used.

6.5 Evaluation of Generated Waiting Strategies and the Influence of Problem Properties

The different waiting strategies as well as the direct policy search, as described in Section 4.4, has been implemented in HeuristicLab. The implementation is based on the generic pickup and delivery simulation optimization model presented in Section 3.1.2. The aim is to evaluate the performance of the different general waiting heuristics as well as the generated specialized policies on a set of benchmark instances with different spatial and temporal characteristics as well as different degrees of dynamism.

The test setup consists of dynamic PDPTW instances that have been proposed by Pankratz (2005) which are derived from the static pickup and delivery instances provided by Li and Lim (2001) which in turn are based on the classical Solomon benchmark instances for the vehicle routing problem with time windows proposed by Solomon (1987). Each of the instances contains 50 dynamically appearing requests.

The 56 different instances are split into different classes which differ in terms of geographical properties and time window properties. In terms of geographical properties, the test set contains clustered, random and mixed locations of transportation requests. The instances also have different time window properties. Some instances have large time windows, others have small time windows. Each instance class represents a combination of these properties, which results in the naming scheme. For example, the instance class C1 contains clustered (C is clustered, R is random, RC is mixed) instances with tight time windows (1 is tight, 2 is large).

Class	Training	Test
C1	5	4
C2	4	4
R1	6	6
R2	5	5
RC1	4	4
RC2	4	4
Total	28	27

Table 6.42: Number of training and test instances for each problem class

The individual classes contain 8-12 problem instances that have similar properties. From each class, half of the instances are used as the test set and the other 4-6 instances as the training set. The training set is used during the

direct policy search and is regarded as historical data to derive knowledge about future demands in the form of stochastic or intensity information. The test set is used to compare the different policies on previously unseen instances. The number of instances for each class is listed in Table 6.42.

In the training phase, no static customers are considered which means that all requests appear dynamically during the planning process. In total, there are 28 training instances in 6 different classes. In the test phase, the different waiting policies are evaluated on the remaining 27 instances with different degree of dynamism ranging from 0% static customers to 90% static customers in steps of 10%. This results in a total of 270 test instances with varying degree of dynamism. Ten independent test runs are performed for each instance. The static customers are chosen randomly according to a uniform distribution for each run.

For each policy, the average values in terms of vehicles, distance, and solution quality are reported over all runs. Additionally, the policies are grouped according to their temporal and spatial characteristics as well as to the degree of dynamism to analyze the influence on the potential savings achieved by anticipatory waiting.

6.5.1 Training and Test Results

In the training phase, the *Stochastic* and *Intensity* policies are parametrized on the 28 training instances as outlined in Section 4.4.3. For route calculation, the push forward insertion heuristic is applied. The results on the training instances are listed in Table 6.43. The specialized policies performed best while the *Variable* policy was the best general heuristic. The *Intensity* policy performed 4.69% while the *Stochastic* policy performed 1.14% better than the *Variable* policy.

The performance of the different policies is evaluated on 270 previously unseen instances in the test phase, that have similar characteristics as the training instances. In the test phase, a tabu search heuristic is applied for route calculation. The average results of the different waiting policies on the test set are listed in Table 6.44. The specialized *Intensity* strategy performed best and achieved 0.70% better results than the second best *Variable* strategy.

To test the results for statistical relevance, a multi-sample statistical analysis was performed where each sample represents the average of the 10 test runs for each of the test instances leading to a sample size of 270 for each waiting policy. After the basic effect of the applied waiting policy on the achieved quality was confirmed (see Table 6.45), a post-hoc analysis was performed with pairwise comparison of the policies (see Table 6.46).

It can be stated that the *Intensity* and *Variable* policies perform significantly better than not applying a waiting policy (*DriveFirst*) while the *Depot* policy performs significantly worse. For all other policies, no significant difference could be identified as opposed to not waiting. In the context of this multiple comparison, it cannot be stated that a single policy performs better than all other policies. However, the *Intensity* and *Variable* strategies perform significantly better than all other policies. In addition to the full pairwise analysis of all policies, a single comparison between these two policies without considering the other policies results in the *Intensity* policy performing better than the *Variable* with a significance of $p = 0.01201$.

	Training Set			
	Distance (km)	Fleet (count)	Costs (Euro)	
DriveFirst	2324.86	22.63	70213.75	
Depot	2343.91	22.96	71232.79	(+1.45%)
WaitFirst	2280.64	22.97	71191.75	(+1.39%)
DW	2294.57	22.84	70805.68	(+0.84%)
Location	2257.18	22.79	70612.74	(+0.57%)
MaxDistance	2296.36	22.50	69785.25	(-0.61%)
ADW	2257.48	22.44	69590.81	(-0.89%)
Distance	2221.29	22.07	68421.29	(-2.55%)
Variable	2170.14	21.60	66970.14	(-4.62%)
Stochastic	2205.86	21.33	66205.86	(-5.71%)
Intensity	2165.72	20.56	63832.39	(-9.09%)

Table 6.43: Average results of the waiting policies in the training phase on the 28 training instances. The relative savings compared to not waiting (DriveFirst) are listed and the policies are ordered according to the achieved quality.

	Test Set			
	Distance (km)	Fleet (count)	Costs (Euro)	
DriveFirst	1980.94	17.89	55659.83	
WaitFirst	1959.53	18.37	57059.53	(+2.51%)
Depot	2005.70	18.31	56947.92	(+2.31%)
DW	1940.37	17.98	55879.25	(+0.39%)
MaxDistance	1937.82	17.92	55683.38	(+0.04%)
Location	1925.70	17.85	55464.59	(-0.35%)
ADW	1931.04	17.76	55223.26	(-0.78%)
Stochastic	1933.76	17.65	54895.98	(-1.37%)
Distance	1909.11	17.53	54493.56	(-2.10%)
Variable	1898.53	17.19	53481.87	(-3.91%)
Intensity	1910.73	17.07	53107.40	(-4.59%)

Table 6.44: Average results of the waiting policies in the test phase on the 270 test instances. The relative savings compared to not waiting (DriveFirst) are listed and the policies are ordered according to the achieved quality.

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_1} = \widetilde{\mu_2} = \widetilde{\mu_3} = \widetilde{\mu_4} = \widetilde{\mu_5} = \widetilde{\mu_6} = \widetilde{\mu_7} = \widetilde{\mu_8} = \widetilde{\mu_9} = \widetilde{\mu_{10}} = \widetilde{\mu_{11}}$	$\neg H_0$	**Quality** of the policies: **DriveFirst** ($\widetilde{x_1} = 52365.72$) **WaitFirst** ($\widetilde{x_2} = 54740.63$) **Depot** ($\widetilde{x_3} = 53981.25$) **DW** ($\widetilde{x_4} = 53933.29$) **Location** ($\widetilde{x_5} = 53321.15$) **MaxDistance** ($\widetilde{x_6} = 53040.49$) **ADW** ($\widetilde{x_7} = 52824.63$) **Distance** ($\widetilde{x_8} = 52694.68$) **Stochastic** ($\widetilde{x_9} = 52205.13$) **Variable** ($\widetilde{x_{10}} = 50697.27$) **Intensity** ($\widetilde{x_{11}} = 50693.84$)	2.2e-16	Reject H_0

Table 6.45: Statistical analysis of the different policies on the 270 test instances. A Friedman test was used to test if any median of the qualities achieved by the policies differ with a significance level of $p < 0.05$.

	ADW	Int-ensity	Depot	Drive-First	Dist-ance	DW	Loc-ation	MaxD-istance	Stoch-astic	Var-iable
Intensity	**2E-16**	-	-	-	-	-	-	-	-	-
Depot	**4E-12**	**2E-16**	-	-	-	-	-	-	-	-
DriveFirst	0.6464	**2E-16**	**7E-11**	-	-	-	-	-	-	-
Distance	**1E-03**	**2E-08**	**6E-13**	0.0589	-	-	-	-	-	-
DW	**2E-11**	**2E-16**	**8E-06**	0.9826	**2E-12**	-	-	-	-	-
Location	0.9826	**7E-16**	**7E-07**	0.9826	**2E-14**	0.5382	-	-	-	-
MaxDistance	**7E-03**	**2E-16**	**1E-06**	0.9826	**2E-16**	0.9826	0.1574	-	-	-
Stochastic	0.9826	**4E-14**	**2E-16**	0.7211	0.9826	**3E-05**	0.136	**4E-03**	-	-
Variable	**2E-16**	0.2902	**2E-16**	**1E-11**	**1E-08**	**2E-16**	**2E-16**	**2E-16**	**8E-07**	-
WaitFirst	**1E-09**	**2E-16**	0.9826	0.0516	**2E-16**	**3E-06**	**2E-16**	**5E-15**	**2E-09**	**2E-16**

Table 6.46: Post-hoc pairwise comparison of the policies on the 270 test instances with a a significance level of $p < 0.05$. The Wilcoxon signed rank test was applied and the p-values were adjusted using Hochberg's procedure. Significant results are highlighted in bold.

6.5.2 Influence of Spatial and Temporal Properties

The test set consisted of instances with different spatial and temporal properties. Concretely, in terms of spatial properties, the performance on instances with clustered (Table 6.47), randomly (Table 6.48), and mixed (Table 6.49) geographically distributed customers while in terms of temporal properties, the performance on instances with tight (Table 6.50) and large (Table 6.51) time windows was evaluated.

The *Intensity* strategy performed best in all comparisons except the mixed instances where the *Variable* strategy performed 0.76% better. For the clustered instances, 11.71% savings were achieved by the best policy compared to 2.24% for the random and 2.27% for the mixed instances. In terms of time windows, the largest savings for instances with small time windows were 3.13% while for large time windows they were 7.14%.

	Clustered Test Instances			
	Distance (km)	Fleet (count)	Costs (Euro)	
DriveFirst	1627.57	16.46	51018.82	
Depot	1621.58	16.36	50709.08	(-0.61%)
Stochastic	1516.06	15.72	48676.06	(-4.59%)
ADW	1520.52	15.65	48466.77	(-5.00%)
DW	1518.53	15.60	48311.03	(-5.31%)
WaitFirst	1495.09	15.23	47170.09	(-7.54%)
MaxDistance	1501.44	15.13	46880.19	(-8.11%)
Location	1476.62	14.98	46420.37	(-9.01%)
Variable	1466.85	14.73	45664.35	(-10.50%)
Distance	1465.20	14.72	45613.95	(-10.59%)
Intensity	1467.71	14.53	45046.46	(-11.71%)

Table 6.47: Average results of the waiting policies on the 80 test instances with geographically clustered customers. The relative savings compared to not waiting (DriveFirst) are listed and the policies are ordered according to the achieved quality.

	Random Test Instances			
	Distance (km)	Fleet (count)	Costs (Euro)	
DriveFirst	1942.07	18.35	56978.43	
WaitFirst	1975.54	19.67	60988.27	(+7.04%)
Location	1949.84	19.19	59533.48	(+4.48%)
MaxDistance	1950.08	19.16	59419.17	(+4.28%)
DW	1941.17	18.93	58722.99	(+3.06%)
Depot	1965.87	18.73	58169.50	(+2.09%)
Distance	1922.24	18.69	57981.34	(+1.76%)
ADW	1928.19	18.59	57687.28	(+1.24%)
Stochastic	1943.62	18.40	57157.26	(+0.31%)
Variable	1907.24	18.19	56490.88	(-0.86%)
Intensity	1913.56	17.93	55703.56	(-2.24%)

Table 6.48: Average results of the waiting policies on the 110 test instances with geographically randomly distributed customers. The relative savings compared to not waiting (DriveFirst) are listed and the policies are ordered according to the achieved quality.

	Mixed Test Instances			
	Distance (km)	Fleet (count)	Costs (Euro)	
DriveFirst	2387.76	18.70	58487.76	
WaitFirst	2401.95	19.72	61546.95	(+5.23%)
Depot	2444.57	19.69	61507.07	(+5.16%)
DW	2361.09	19.06	59537.34	(+1.79%)
MaxDistance	2357.35	19.00	59349.85	(+1.47%)
Location	2341.58	18.86	58914.08	(+0.73%)
ADW	2345.48	18.75	58591.73	(+0.18%)
Distance	2334.97	18.75	58577.47	(+0.15%)
Stochastic	2337.89	18.56	58006.64	(-0.82%)
Intensity	2349.87	18.42	57598.62	(-1.52%)
Variable	2318.24	18.28	57161.99	(-2.27%)

Table 6.49: Average results of the waiting policies on the 80 test instances with both geographically randomly distributed and clustered customers. The relative savings compared to not waiting (DriveFirst) are listed and the policies are ordered according to the achieved quality.

	Small TW Test Instances			
	Distance (km)	Fleet (count)	Costs (Euro)	
DriveFirst	1918.63	22.19	68473.63	
WaitFirst	1974.66	23.29	71844.66	(+4.92%)
Depot	1965.70	23.00	70961.42	(+3.63%)
DW	1935.39	22.58	69668.96	(+1.75%)
MaxDistance	1927.40	22.52	69489.54	(+1.48%)
Location	1924.77	22.42	69186.91	(+1.04%)
Stochastic	1929.94	22.37	69039.94	(+0.83%)
ADW	1917.60	22.31	68836.89	(+0.53%)
Distance	1908.63	22.16	68386.49	(-0.13%)
Variable	1883.39	21.59	66644.82	(-2.67%)
Intensity	1883.23	21.48	66327.52	(-3.13%)

Table 6.50: Average results of the waiting policies on the 140 test instances with small time windows. The relative savings compared to not waiting (DriveFirst) are listed and the policies are ordered according to the achieved quality.

	Large TW Test Instances			
	Distance (km)	Fleet (count)	Costs (Euro)	
DriveFirst	2048.04	13.27	41860.35	
Depot	2048.76	13.27	41856.46	(-0.01%)
WaitFirst	1943.23	13.06	41137.08	(-1.73%)
DW	1945.73	13.03	41028.81	(-1.99%)
MaxDistance	1949.05	12.96	40815.21	(-2.50%)
Location	1926.71	12.92	40686.71	(-2.80%)
ADW	1945.51	12.87	40562.44	(-3.10%)
Stochastic	1937.87	12.58	39664.02	(-5.25%)
Distance	1909.63	12.54	39531.94	(-5.56%)
Variable	1914.84	12.46	39306.38	(-6.10%)
Intensity	1940.35	12.31	38870.35	(-7.14%)

Table 6.51: Average results of the waiting policies on the 130 test instances with large time windows. The relative savings compared to not waiting (DriveFirst) are listed and the policies are ordered according to the achieved quality.

6.5.3 Influence of Degree of Dynamism

The test set contains instances with different degree of dynamism ranging from 10% to 100% dynamic customers. The performance of the four best waiting policies depending on the degree of dynamism is illustrated in Figure 6.2. The *Intensity* policy performs best in all degree of dynamism except the lowest one with 10% dynamic customers where the *Variable* policy performs 0.19% better. The largest potential savings of 5.82% compared to not waiting (*DriveFirst*) could be achieved with 60% static customers.

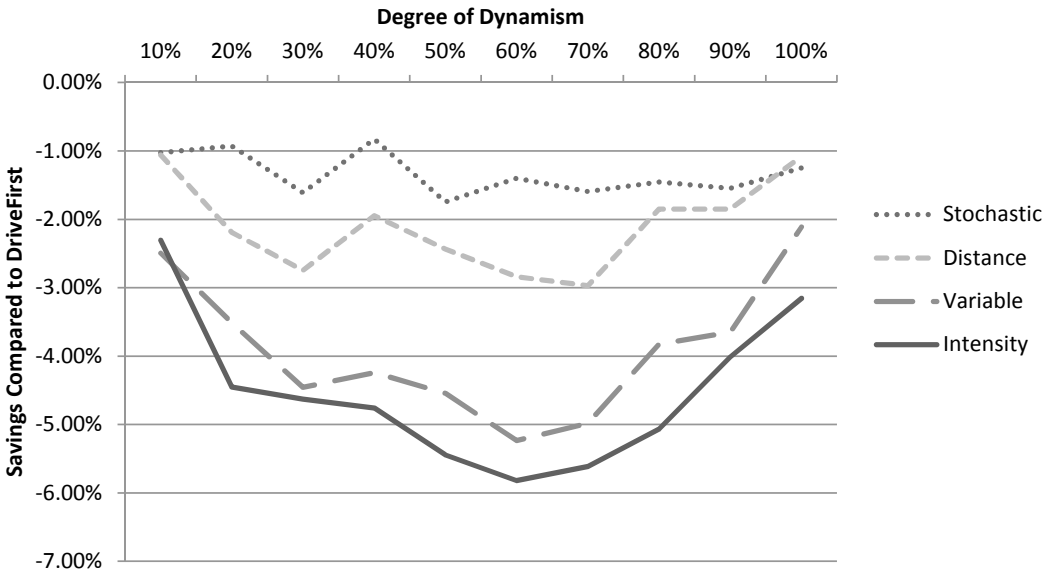

Figure 6.2: Savings of the four best waiting policies compared to not waiting (DriveFirst) on the test set containing instances with varying degree of dynamism. Each category (10%-100% dynamic customers) contains 27 test instances.

6.6 Benefits of Situational Selection in Environments with Changing Uncertainty

The methodology presented in Section 5.2 has been implemented in HeuristicLab combining all algorithmic concepts of the portfolio approach presented in Section 5.1 into a single optimization environment.

As described in Section 5.2, the multiple scenario algorithm (*MSA*), the waiting strategies presented in Section 4.4 (*AW*), as well as a purely reactive planning (*AdHoc*) are combined to a portfolio. Different weighting functions are evaluated for the algorithm portfolio which weigh near-term requests differently than long-term requests. Portfolios with uniform (*PfUniform*), linear (*PfLinear*), and exponential (*PfExpα*) weighting functions are investigated. In terms of exponential weighting, α values of 0.1, 0.5, 1.0, 1.5, and 2.0 are considered.

The individual heuristics as well as portfolios with different weighting functions are evaluated on stochastic and dynamic pickup and delivery problem instances. The instances have been derived from the instances provided by Pankratz (2005) which have been also used in the previous section. However, only instances with 100% degree of dynamism are considered. A total of 27 test instances split in 6 classes are used with clustered (C_), random (R_), and mixed (RC_) geographically distributed requests as well small (_1) and large (_2) time windows. It is feasible to use only the test instances as presented in the previous section, since the waiting strategies have been generated on the remaining 28 training instances which could influence the results.

Two benchmark sets are derived from these 27 test instances by generating an appearance probability between 10% and 90% for each request. An evaluation benchmark set is generated with fixed appearance probabilities which is used during the portfolio design phase (as presented in Section 5.2.3) while a test set with changing appearance probabilities is generated for evaluating the resulting portfolio (as presented in this section).

For the evaluation set, a fixed appearance probability ranging from 10% to 90% in steps of 10% was considered for each of the 6 test set classes leading to a total of $6 * 9 = 54$ evaluation classes. Since each class consists of 27 instances, the evaluation set consists of a total of 1458 instances. The evaluation set was solely used in the portfolio design phase to evaluate the robustness of the individual policies and to draw conclusions about the performance profile.

After portfolio design, a set of test instances with changing uncertainty was generated based on the 27 original test instances. The planning horizon

is split into three phases for each instance. In the first third of the horizon, the requests appear with a probability of 90% while in the second third, the probability is 50%, and in the last third it is 10%. The data quality thus changes over time. Based on these 27 test instances with changing uncertainty, the portfolio was compared to the individual policies and different weighting functions were evaluated. From each of the 27 stochastic instances, 20 fixed samples were created leading to a total of 540 dependent test runs for each policy.

The overall results are listed in Table 6.52. The MSA was the individual policy and servers as a baseline for the portfolio as well as the other policies. The exponential weighting with an α value of 1.5 was the best portfolio parametrization and achieved a 2.16% saving compared to the MSA.

The results were tested for statistical significance. As listed in Table 6.53, the used policy has a significant effect on the achieved quality. In a detailed pairwise comparison that is listed in Table 6.54 it can be observed, that the portfolio approach significantly outperforms the individual algorithms. The different weighting parametrization values cannot be significantly distinguished, except the best (*PFExp15*) and the worst (*PFExp01*).

	Overall Results			
	Distance (km)	Fleet (count)	Costs (Euro)	
MSA	1760.36	12.53	39338.14	
AdHoc	1369.13	14.53	44969.13	(+14.31%)
AW	1302.07	13.79	42657.63	(+8.44%)
PfExp01	1652.81	12.46	39030.58	(-0.78%)
PfUniform	1573.20	12.44	38906.54	(-1.10%)
PfLinear	1597.24	12.36	38675.02	(-1.69%)
PfExp20	1590.28	12.36	38668.05	(-1.70%)
PfExp10	1590.40	12.32	38551.51	(-2.00%)
PfExp05	1596.18	12.32	38546.18	(-2.01%)
PfExp15	1580.92	12.30	38486.47	(-2.16%)

Table 6.52: Average results of the policies on the overall set of test instances with changing degree of uncertainty. The portfolios with different weighting functions are listed with a *Pf* prefix. The results are compared to the overall best individual heuristic (MSA).

H_0	H_a	Samples	p-Value	Result
$\widetilde{\mu_1} = \widetilde{\mu_2} = \widetilde{\mu_3} = \widetilde{\mu_4} = \widetilde{\mu_5} = \widetilde{\mu_6} = \widetilde{\mu_7} = \widetilde{\mu_8} = \widetilde{\mu_9} = \widetilde{\mu_{10}}$	$\neg H_0$	**Quality** of the policies: **AdHoc** ($\widetilde{x_1} = 42977.33$) **AW** ($\widetilde{x_2} = 40299.32$) **MSA** ($\widetilde{x_3} = 37710.34$) **PfLinear** ($\widetilde{x_4} = 37068.85$) **PfUniform** ($\widetilde{x_5} = 37160.09$) **PfExp01** ($\widetilde{x_6} = 37295.69$) **PfExp05** ($\widetilde{x_7} = 37109.65$) **PfExp10** ($\widetilde{x_8} = 36916.71$) **PfExp15** ($\widetilde{x_9} = 37076.29$) **PfExp20** ($\widetilde{x_{10}} = 37169.94$)	2.2e-16	Reject H_0

Table 6.53: Statistical analysis of the different policies on the 540 test runs. A Friedman test was used to test if any median of the qualities achieved by the policies differ with a significance level of $p < 0.05$.

	AdHoc	AW	MSA	PfExp01	PfExp05	PfExp10	PfExp15	PfExp20	PfLinear
AW	**7E-16**	-	-	-	-	-	-	-	-
MSA	**2E-16**	**4E-14**	-	-	-	-	-	-	-
PfExp01	**2E-16**	**2E-16**	**0.37078**	-	-	-	-	-	-
PfExp05	**2E-16**	**2E-16**	**1E-04**	0.12315	-	-	-	-	-
PfExp10	**2E-16**	**2E-16**	**2E-04**	0.06321	0.86264	-	-	-	-
PfExp15	**2E-16**	**2E-16**	**5E-05**	**0.02614**	0.86264	0.86264	-	-	-
PfExp20	**2E-16**	**2E-16**	**7E-04**	0.28216	0.86264	0.86264	0.86264	-	-
PfLinear	**2E-16**	**2E-16**	**1E-03**	0.60437	0.86264	0.86264	0.86264	0.86264	-
PfUniform	**2E-16**	**2E-16**	**0.05785**	0.86264	0.86264	0.86264	0.86264	0.86264	0.86264

Table 6.54: Post-hoc pairwise comparison of the policies on the 540 test runs with a a significance level of $p < 0.05$. The Wilcoxon signed rank test was applied and the p-values were adjusted using Hochberg's procedure. Significant results are highlighted in bold.

6.6.1 Detailed Results per Instance Class

Since the performance profiles presented in Section 5.2 differ quite strongly for the different instance classes, a detailed consideration of the results for each class makes sense. Again, the results are compared to the best individual policy which is the MSA.

Results for the clustered instances with small time windows (C1) are listed in Table 6.55. The Portfolio approaches cannot outperform the best individual heuristic (AW) in that case. A similar picture is drawn for the clustered instances with large time windows (C2) as listed in Table 6.56 while the AW clearly outperforms the best overall heuristic which is the MSA.

The results for the random instances with small time windows (R1) are listed in Table 6.57 and the instances with large time windows (R2) in Table 6.58. Results for the mixed instances with small time windows (RC1) are presented in Table 6.59 and the instances with large time windows (RC2) are listed in Table 6.60. For the instances other than C1 and C2, the situation changes and the portfolio approach gains a clear advantage compared to the best individual heuristic.

In general, the *PfExp15* is the best weighting strategy especially for large time windows. However, in instances with small time windows, the *PfExp05* strategy outperforms the *PfExp15* strategy by 0.36%. For large time windows, the *PfExp15* has an advantage of 1.57% compared to the *PfExp05* strategy leading to better overall results.

The results illustrate the importance of a differentiated view in terms of problem characteristics. When selecting the best individual policy per instance class (AW for the clustered and MSA for the remaining instances), the advantage of the *PfExp15* portfolio approach decreases to of 0.63%. If the best weighting function is selected per class (*Linear* for C1 and C2, *PfExp05* for R1 and RC1, and *PfExp15* for R2 and RC2), the advantage of the dynamic portfolio compared to a static selection based on the instance class increases to 1.06%.

	Class C1			
	Distance (km)	Fleet (count)	Costs (Euro)	
MSA	1551.02	19.39	59713.52	
AdHoc	1448.74	20.29	62311.24	(+4.35%)
PfExp01	1407.07	18.65	57357.07	(-3.95%)
AW	1365.74	18.61	57203.24	(-4.20%)
PfExp20	1365.74	18.61	57203.24	(-4.20%)
PfExp05	1365.74	18.61	57203.24	(-4.20%)
PfExp15	1365.74	18.61	57203.24	(-4.20%)
PfUniform	1365.73	18.61	57203.23	(-4.20%)
PfExp10	1365.72	18.61	57203.22	(-4.20%)
PfLinear	1365.69	18.61	57203.19	(-4.20%)

Table 6.55: Average results of the policies on the C1 instances with changing degree of uncertainty. The portfolios with different weighting functions are listed with a *Pf* prefix. The results are compared to the overall best single-algorithm heuristic (MSA).

	Class C2			
	Distance (km)	Fleet (count)	Costs (Euro)	
MSA	1560.94	8.64	27473.44	
AdHoc	1070.03	9.60	29870.03	(+8.72%)
PfExp01	1087.63	9.29	28950.13	(+5.37%)
AW	881.53	8.34	25894.03	(-5.75%)
PfExp15	878.89	8.31	25816.39	(-6.03%)
PfExp20	878.56	8.31	25816.06	(-6.03%)
PfExp10	878.48	8.31	25815.98	(-6.03%)
PfExp05	878.48	8.31	25815.98	(-6.03%)
PfUniform	878.48	8.31	25815.98	(-6.03%)
PfLinear	879.08	8.29	25741.58	(-6.30%)

Table 6.56: Average results of the policies on the C2 instances with changing degree of uncertainty. The portfolios with different weighting functions are listed with a *Pf* prefix. The results are compared to the overall best single-algorithm heuristic (MSA).

	Class R1			
	Distance (km)	Fleet (count)	Costs (Euro)	
MSA	1266.75	13.30	41166.75	
AdHoc	1118.06	14.80	45518.06	(+10.57%)
AW	1112.93	14.63	44987.93	(+9.28%)
PfUniform	1247.77	13.52	41797.77	(+1.53%)
PfExp20	1244.49	13.38	41369.49	(+0.49%)
PfLinear	1253.29	13.34	41278.29	(+0.27%)
PfExp15	1236.63	13.27	41046.63	(-0.29%)
PfExp01	1244.16	13.25	40994.16	(-0.42%)
PfExp10	1240.48	13.24	40965.48	(-0.49%)
PfExp05	1234.84	13.21	40859.84	(-0.75%)

Table 6.57: Average results of the policies on the R1 instances with changing degree of uncertainty. The portfolios with different weighting functions are listed with a *Pf* prefix. The results are compared to the overall best single-algorithm heuristic (MSA).

	Class R2			
	Distance (km)	Fleet (count)	Costs (Euro)	
MSA	2048.40	8.21	26678.40	
AdHoc	1175.69	10.60	32975.69	(+23.60%)
AW	1103.69	9.44	29423.69	(+10.29%)
PfUniform	1803.67	8.55	27453.67	(+2.91%)
PfExp05	1930.62	8.33	26920.62	(+0.91%)
PfExp20	1884.93	8.33	26874.93	(+0.74%)
PfLinear	1908.13	8.29	26778.13	(+0.37%)
PfExp01	2007.53	8.24	26727.53	(+0.18%)
PfExp10	1883.10	8.17	26393.10	(-1.07%)
PfExp15	1876.62	8.12	26236.62	(-1.66%)

Table 6.58: Average results of the policies on the R2 instances with changing degree of uncertainty. The portfolios with different weighting functions are listed with a *Pf* prefix. The results are compared to the overall best single-algorithm heuristic (MSA).

	Class RC1			
	Distance (km)	Fleet (count)	Costs (Euro)	
MSA	1724.29	16.36	50811.79	
AW	1837.85	20.55	63487.85	(+24.95%)
AdHoc	1831.92	20.55	63481.92	(+24.94%)
PfExp10	1741.37	16.29	50603.87	(-0.41%)
PfUniform	1720.81	16.19	50283.31	(-1.04%)
PfExp01	1729.43	16.16	50216.93	(-1.17%)
PfExp15	1701.85	16.15	50151.85	(-1.30%)
PfLinear	1728.12	16.13	50103.12	(-1.39%)
PfExp20	1716.66	16.09	49979.16	(-1.64%)
PfExp05	1712.14	16.04	49824.64	(-1.94%)

Table 6.59: Average results of the policies on the RC1 instances with changing degree of uncertainty. The portfolios with different weighting functions are listed with a *Pf* prefix. The results are compared to the overall best single-algorithm heuristic (MSA).

	Class RC2			
	Distance (km)	Fleet (count)	Costs (Euro)	
MSA	2585.55	9.95	32435.55	
AdHoc	1744.20	12.21	38381.70	(+18.33%)
AW	1654.85	11.81	37092.35	(+14.36%)
PfLinear	2543.41	10.03	32618.41	(+0.56%)
PfExp05	2552.35	9.95	32402.35	(-0.10%)
PfExp20	2550.51	9.94	32363.01	(-0.22%)
PfUniform	2527.86	9.93	32302.86	(-0.41%)
PfExp10	2535.00	9.88	32160.00	(-0.85%)
PfExp01	2556.66	9.83	32031.66	(-1.25%)
PfExp15	2523.98	9.76	31811.48	(-1.92%)

Table 6.60: Average results of the policies on the RC2 instances with changing degree of uncertainty. The portfolios with different weighting functions are listed with a *Pf* prefix. The results are compared to the overall best single-algorithm heuristic (MSA).

Chapter 7

Conclusions and Outlook

The dynamic vehicle routing problem remains a challenging combinatorial optimization problem which requires specialized algorithmic approaches incorporating knowledge about the problem structure and future events. When developing algorithmic strategies, a compromise has to be made between robustness and specialization according to the *no free lunch* theorem (Wolpert and Macready, 1997). This thesis has investigated research directions aiming to overcome this issue by presenting a methodology which combines ideas from several related areas with the main vision of a decision support system that semi-automatically adapts its algorithmic strategies.

7.1 Summary of Main Achievements

The main step beyond the current state-of-the art made in this thesis is the semi-automated adaption of heuristic methods for dynamic vehicle routing problems considering potentially changing problem structures and characteristics. By combining several specialized heuristics to a portfolio, the presented algorithmic framework enables a specialization to the problem structure while being robust in terms of changing characteristics.

Three essential building blocks of an adaptive algorithmic framework for dynamic vehicle routing have been identified while these topics are closely related. Rich variants of vehicle routing problems are modeled using simulation capturing the dynamic interactions and stochastic influence factors. Based on a simulation model and training data, specialized policies are generated in a semi-automated way. Multiple specialized policies are combined to an algorithm portfolio and selected according to the problem characteristics. These building blocks enable the modeling of rich variants and the semi-automated adaption of the algorithmic strategies on a meta-level.

7.1.1 Simulation Optimization of Production and Logistic Scenarios

A generic simulation optimization modeling framework for dynamic vehicle routing problems has been presented with two specializations for pickup and delivery and inventory routing problems. The simulation model captures the dynamic interactions as well as the stochastic influence factors and serves as a basis for manual and semi-automated algorithm design. A realistic problem model with rich side-constraints enables the transfer of the findings back into practice. In particular, three practical case-studies of production and logistic scenarios have been presented and the findings can be summarized as following:

- The simulation optimization of the transport activities within cold charge steel production based on a scenario of a steel plant has yielded valuable insights in terms of potential improvements of the slab handling process. The stacking and shuffling operations have been identified as a main bottleneck. Additionally, dynamically arriving information from upstream and downstream processes has a significant effect on the solution quality. As a result, an integrated optimization approach should be investigated combining scheduling, transport, and storage assignment.

- The integrated optimization of warehousing and transport activities based on a scenario of one of the worlds largest supplier of firefighting vehicles has been investigated. For that purpose, a warehousing simulation, a storage assignment optimization, and a routing simulation optimization model have been coupled to study the interrelations between the individual sub-activities within material handling. The picking schedule influences the efficiency of the storage assignment strategy as well as the in-house transport and is thus of major importance. By considering the down-stream transport to the workstations already while picking the items and accepting a slight decrease in the efficiency of the warehouse activities, an overall increase of the material handling efficiency could be observed. This again highlights the importance of integrated optimization approaches.

- A simulation-based optimization approach has been applied based on an inventory routing scenario of an Austrian retailer who delivers fast-moving consumer goods to supermarkets. A novel aspect has been the consideration of mixed scenarios where only part of the customers are switched to a vendor-managed inventory while the other customers

retain a threshold-based order strategy. The application of simulation optimization as a scenario technique has yielded insights in terms of exogenous and endogenous influence factors on the efficiency of a vendor-managed inventory. The main savings potential lies in smoothing fluctuating demand patterns by distributing replenishments more equally and as a result lowering the peak resource utilization. The simulation-based sensitivity analysis has revealed that the service quality and the demand distribution have the largest impact on the potential savings.

7.1.2 Algorithmic Generation of Specialized Routing Policies

An algorithmic framework for the semi-automated generation of specialized policies using a black-box simulation model and training data has been proposed. Heuristic policies have been automatically adapted and generated based on direct policy search and reinforcement learning. The semi-automated generation of policies combines the ability of human experts to define algorithmic building blocks and the robustness of evolutionary algorithms to adapt and assemble them in superior ways. Routing policies have been generated for different variants of dynamic vehicle routing problems and their potential to perform well on previously unseen instances that have similar characteristics as the training data has been confirmed. This ability of a semi-automated specialization to problem characteristics and to generate re-usable heuristics in an offline training phase is an essential building block of a system adapting to its problem environment. Important design decisions are the policy representation, the learning approach, and the search algorithm. The generation of three particular routing policies embedded in a heuristic framework has been investigated:

- Replenishment policies for inventory routing scenarios have been generated that are specialized to different scenarios. By integrating the refill policies into a heuristic framework, the inherent problem complexity of the high-dimensional scenario, which has been derived from real-world data, could be mitigated. The reduction in complexity has been achieved by combining several approximations. A two-stage approach has been used to divide the inventory replenishment from the routing decisions. The inventory replenishment decisions are based on abstract features that have been defined by a domain expert which are combined to a parametrized policy. The study clearly illustrated the potential of applying direct policy search and integrating the resulting policies in a heuristic framework to optimize highly complex

scenarios. Furthermore, an investigation of local optima has been presented indicating a multi-modal and globally non-convex structure of the parameter landscape.

- Dispatching policies for dial-a-ride problems have been evolved that allow an autonomous operation of vehicles in an agent-based simulation environment. The evolved policies have been interpreted and the most impactful domain features have been identified which allows a feedback loop to the policy design phase. Two different policy representations have been compared while a tree representation has performed slightly better than a linear representation. Compared to a planning approach, the dispatching policies have performed poorer in low intensity while they performed comparably well in high intensity scenarios. In both cases, they have required a significantly less runtime during online operation. The computational effort is transferred to the offline training phase. These findings indicate, that specialized dispatching policies are especially applicable in highly volatile environments where a reactive acting and a quick response time is required.

- Waiting policies for pickup and delivery problems have been generated with the aim of anticipating future requests by distributing the waiting time based on problem characteristics. The generated specialized policies have outperformed the general heuristics that are designed to work well regarding a large range of problem characteristics. However, spatial and temporal problem characteristics have a large impact on the savings potential of anticipatory waiting. Especially geographical clustering of requests and large time windows have been identified as beneficial properties. Applying specialized waiting strategies is beneficial in cases where the potential savings are comparably large where additional optimization potential compared to general heuristics can be exploited. These results clearly illustrate the need for combining different solution methods based on the problem characteristics.

7.1.3 Dynamic Situational Selection of Routing Policies

An algorithm selection framework based on a portfolio of specialized policies has been presented combining all methodological developments of this thesis towards the goal of adaptive decision support systems. By integrating algorithm selection and simulation optimization, a dynamic situational selection of specialized routing policies has been achieved. Two possible extensions

with learning capabilities have been presented for automated portfolio design and building knowledge about the mapping between problem characteristics and algorithm performance. Three important design decisions have been identified for applying the framework to a particular application area. The problem characterization, the portfolio design, and the problem to algorithm mapping aspects are specialized based on the generic framework. A particular case-study of an environment with changing characteristics has been presented to illustrate the potential of the methodology:

- A dynamic pickup and delivery problem with stochastic requests has been investigated where the appearance probabilities of the requests changes over time. In a robustness analysis it has been identified, that the performance of the individual policies depends on spatial and temporal characteristics of the problem instance as well as the uncertainty in terms of the appearance of requests. Furthermore, it has been observed that the policies complement each other and have individual strengths and weaknesses depending on these problem characteristics. These insights were incorporated in the portfolio design combining several specialized heuristics and deriving a set of selection rules. In particular, semi-automatically generated waiting policies have been combined with human-designed heuristics known from the literature. The portfolio approach has clearly outperformed each individual heuristic on the overall instance set. The policies have complemented each other especially in terms of spatial and temporal characteristics and the dynamic selection has revealed additional optimization potential. The highly specialized generated policies have complemented the human-designed heuristics which are designed for robust behavior in a large number of cases. By combining several policies to a portfolio a better performance could by achieved than by each individual heuristic.

7.2 Research Directions

Important algorithmic building blocks of a decision support system that is adaptive in terms of problem characteristics on a meta-level have been highlighted in this thesis. While the potential of the methodology has been illustrated and a semi-automated adaption of algorithmic strategies has been accomplished, there are several open issues to reach the goal of a self-adaptive decision support system which autonomously learns within its routing environment:

- It is clear that the presented case-study for a portfolio-based algorithm selection merely scratched the surface of the possibilities the methodological framework offers. Several research directions that are important for automated algorithm selection have been outlined including fitness landscape analysis for problem characterization, policy generation for portfolio design, machine learning for mapping problem characteristics to the algorithm space, and selecting multiple policies:

  In terms of problem characterization, the investigation of fitness landscapes of dynamic vehicle routing problem is an open issue. Few work exists on fitness landscape analysis for dynamic problems (such as Richter (2009)) and it is certainty an interesting research direction to investigate the fitness landscape for dynamic vehicle routing problems. New insights on the problem structure could lead to a better understanding about the links between problem characteristics and algorithm performance.

  Another potentially fruitful direction is to apply machine learning within online systems. One possibility for integration is to incorporate an automated portfolio extension element that explores new unseen problem characteristics and automatically extends the portfolio with newly generated policies. Another possibility is a learning element which derives knowledge on the mapping between problem characteristics and algorithm performance based on feedback during operation and constantly adapts the selection rules.

  While in the presented case-study only a single algorithm running on a single computer has been selected, in principle multiple policies can be selected at once to be executed in parallel. This would allow the utilization of parallel computing infrastructures such as clusters or multi-core processors. It would be required to derive mapping knowledge about a set of selected algorithms which are executed in parallel.

- The methodology presented in this thesis worked on the meta-level by generating policies and selecting among them. One open issue is the incorporation of self-adaptive behavior within the policies itself on the algorithm level. A fruitful research direction to be integrated into the methodological framework is the investigation of metaheuristics in dynamic environments (Alba et al., 2012). An example is adapting certain algorithm parameters such as the mutation rate when the environment is changing.

- The interpretation of the evolved policies remains a difficult issue even though the proposed methodology is based on white-box techniques.

A promising research direction towards the understanding of the structure of the policy space is fitness landscape analysis revealing features such as plateaus, ruggedness, and distribution of local optima. The findings regarding the meta-level landscape can be linked to the fitness landscape of the underlying optimization problem in terms of common features as highlighted by Burke et al. (2013). A preliminary study investigating the properties of parameter landscapes has been presented in Section 4.2.3 including only a rather small number of local optima. Advanced fitness landscape analysis methods require the exploration of a multitude of solution candidates leading to a high computational complexity since each evaluation requires a simulation run. Cloud computing resources could be utilized to perform more extensive studies of the policy space.

- A large potential has been identified in the presented case studies in production and logistics in terms of integrated problem formulations. It was illustrated, that the combination of multiple optimization and simulation components can provide a holistic view on the whole process. In that context, an adaptive algorithmic framework for dynamic vehicle routing could be an important building block of such an optimization network. Whenever decisions about related activities such as storage assignment or scheduling are made, the routing environment might change requiring a constant adaption to the new conditions.

Summarizing, the potential of policy generation and algorithm selection performed on a meta-level based on a realistic simulation model has been clearly identified in the presented case studies. Raising the abstraction to automatically generated and adapting portfolios bears optimization potential compared to human-designed algorithms. However, to reach the vision of a self-adaptive decision support system that autonomously learns and adapts in changing environments, a fundamental understanding of problem characteristics as well as the incorporation of machine learning is required which leads to interesting research directions based on the presented algorithmic framework.

Bibliography

Affenzeller, M., Wagner, S., Winkler, S., and Beham, A. (2009). *Genetic algorithms and genetic programming: modern concepts and practical applications*. CRC Press.

Alba, E., Nakib, A., and Siarry, P. (2012). *Metaheuristics for dynamic optimization*. Springer Publishing Company, Incorporated.

Altinkemer, K. and Gavish, B. (1991). Parallel savings based heuristics for the delivery problem. *Operations Research*, 39(3):456–469.

Andersson, H., Hoff, A., Christiansen, M., Hasle, G., and Løkketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9):1515–1536.

Angelelli, E., Grazia Speranza, M., and Savelsbergh, M. W. (2007). Competitive analysis for dynamic multiperiod uncapacitated routing problems. *Networks*, 49(4):308–317.

Archetti, C. and Speranza, M. G. (2008). The split delivery vehicle routing problem: A survey. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 103–122. Springer.

Attanasio, A., Bregman, J., Ghiani, G., and Manni, E. (2007). Real-time fleet management at ecourier ltd. In *Dynamic Fleet Management*, pages 219–238. Springer.

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

Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., and Talamo, M. (2001). Algorithms for the on-line travelling salesman 1. *Algorithmica*, 29(4):560–581.

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

Bäck, T., Hoffmeister, F., and Schwefel, H.-P. (1991). A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Citeseer.

Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, 59(5):1269–1283.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

Balinski, M. L. and Quandt, R. E. (1964). On an integer program for a delivery problem. *Operations Research*, 12(2):300–304.

Barceló, J., Grzybowska, H., and Pardo, S. (2007). Vehicle routing and scheduling models, simulation and city logistics. In *Dynamic Fleet Management*, pages 163–195. Springer.

Beaudry, A., Laporte, G., Melo, T., and Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR spectrum*, 32(1):77–107.

Beck, J. C. and Freuder, E. C. (2004). Simple rules for low-knowledge algorithm selection. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 50–64. Springer.

Beham, A., Affenzeller, M., Wagner, S., and Kronberger, G. (2008). Simulation optimization with heuristiclab. In *20th European Modeling and Simulation Symposium EMSS 2008*. IMMM.

Beham, A., Kofler, M., Affenzeller, M., and Wagner, S. (2009a). Evolutionary selection in simulation-based optimization. In *Computer Aided Systems Theory-EUROCAST 2009*, pages 761–768. Springer.

Beham, A., Kofler, M., Wagner, S., and Affenzeller, M. (2009b). Agent-based simulation of dispatching rules in dynamic pickup and delivery problems. In *Logistics and Industrial Informatics, 2009. LINDI 2009. 2nd International*, pages 1–6. IEEE.

Beham, A., Kofler, M., Wagner, S., and Affenzeller, M. (2009c). Coupling simulation with heuristiclab to solve facility layout problems. In *Winter Simulation Conference*, pages 2205–2217. Winter Simulation Conference.

Beham, A., Pitzer, E., Wagner, S., Affenzeller, M., Altendorfer, K., Felberbauer, T., and Bäck, M. (2012). Integration of flexible interfaces in optimization software frameworks for simulation-based optimization. In *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, GECCO Companion '12, pages 125–132, New York, NY, USA. ACM.

Bell, W., Dalberto, L., Fisher, M., Greenfield, A., Jaikumar, R., Kedia, P., Mack, R., and Prutzman, P. (1983). Improving the distribution of industrial gases with an online computerized routing and scheduling optimizer. *Interfaces*, 13:4–23.

Bent, R. and Van Hentenryck, P. (2004a). Regrets only! online stochastic optimization under time constraints. In *AAAI*, volume 4, pages 501–506.

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

Bent, R. W. and Van Hentenryck, P. (2004b). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987.

Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15.

Bertazzi, L., Savelsbergh, M., and Speranza, M. (2007). Inventory routing. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges, Operations Research/Computer Science Interfaces Serices*, volume 43, pages 49–72. Springer.

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

Bertsimas, D. J. and Van Ryzin, G. (1993). Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles. *Operations Research*, 41(1):60–76.

Bieding, T., Görtz, S., and Klose, A. (2009). On line routing per mobile phone a case on subsequent deliveries of newspapers. In *Innovations in Distribution Logistics*, pages 29–51. Springer.

Bodin, L. D. (1975). A taxonomic structure for vehicle routing and scheduling problems. *Computers & Urban Society*, 1(1):11–29.

Branke, J., Middendorf, M., Noeth, G., and Dessouky, M. (2005). Waiting strategies for dynamic vehicle routing. *Transportation Science*, 39:298–312.

Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118.

Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., and Woodward, J. R. (2009). Exploring hyper-heuristic methodologies with genetic programming. In *Computational intelligence*, pages 177–201. Springer.

Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*. CRC Press.

Campbell, A. M. and Savelsbergh, M. W. (2004). A decomposition approach for the inventory-routing problem. *Transportation Science*, 38(4):488–502.

Can, B., Beham, A., and Heavey, C. (2008). A comparative study of genetic algorithm components in simulation-based optimisation. In *Proceedings of the 40th Conference on Winter Simulation*, pages 1829–1837. Winter Simulation Conference.

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

Chitty, D. M. and Hernandez, M. L. (2004). A hybrid ant colony optimisation technique for dynamic vehicle routing. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 48–59. Springer.

Choi, E. and Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7):2080–2095.

Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *OR*, pages 309–318.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. (2002). Vrp with time windows. *The vehicle routing problem*, 9:157–193.

Cordeau, J.-F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, pages 928–936.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Chapter 6 vehicle routing. In Barnhart, C. and Laporte, G., editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367 – 428. Elsevier.

Crainic, T. G. and Toulouse, M. (2003). *Parallel strategies for meta-heuristics*. Springer.

Crainic, T. G., Toulouse, M., and Gendreau, M. (1997). Toward a taxonomy of parallel tabu search heuristics. *INFORMS Journal on Computing*, 9(1):61–72.

Cruz-Reyes, L., Gómez-Santillán, C., Pérez-Ortega, J., Landero, V., Quiroz, M., and Ochoa, A. (2012). Algorithm selection: From meta-learning to hyper-heuristics. In Koleshko, V. M., editor, *Intelligent Systems*, pages 77–102. InTech.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

De Franceschi, R., Fischetti, M., and Toth, P. (2006). A new ilp-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2-3):471–499.

Desrochers, M., Lenstra, J. K., and Savelsbergh, M. W. (1990). A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46(3):322–332.

Doerner, K. F. and Schmid, V. (2010). Survey: matheuristics for rich vehicle routing problems. In *Hybrid Metaheuristics*, pages 206–221. Springer.

Dohn, A. and Clausen, J. (2010). Optimising the slab yard planning and crane scheduling problem using a two-stage heuristic. *International Journal of Production Research*, 48(15):4585–4608.

Drexl, M. (2012). Rich vehicle routing in theory and practice. *Logistics Research*, 5(1-2):47–63.

Eiben, A. E. and Smith, J. E. (2010). *Introduction to evolutionary computing*, volume 2. Springer Berlin.

Eilon, S., Watson-Gandy, C. D. T., and Christofides, N. (1971). *Distribution management: mathematical modelling and practical analysis*. Griffin London.

Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483.

Fawcett, T. E. and Utgoff, P. E. (1992). Automatic feature generation for problem solving systems. In *ML*, pages 144–153. Citeseer.

Ferrucci, F., Bock, S., and Gendreau, M. (2012). A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. *European Journal of Operational Research*.

Fisher, H. and Thompson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, pages 225–251.

Fisher, M. L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124.

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

Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). The period vehicle routing problem and its extensions. In *The vehicle routing problem: latest advances and new challenges*, pages 73–102. Springer.

Fu, M. C., Glover, F. W., and April, J. (2005). Simulation optimization: a review, new developments, and applications. In *Simulation Conference, 2005 Proceedings of the Winter*, pages 13–pp. IEEE.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511.

Gambardella, L. M., Rizzoli, A. E., Oliverio, F., Casagrande, N., Donati, A. V., Montemanni, R., and Lucibello, E. (2003). Ant colony optimization for vehicle routing in advanced logistics systems. In *Proceedings of the International Workshop on Modelling and Applied Simulation (MAS 2003)*, pages 3–9.

García, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617–644.

Garvin, W., Crandall, H., John, J., and Spellman, R. (1957). Applications of linear programming in the oil industry. *Management Science*, 3(4):407–430.

Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science*, 33(4):381–390.

Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290.

Gendreau, M., Laporte, G., and Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12.

Gendreau, M., Laporte, G., and Semet, F. (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel computing*, 27(12):1641–1653.

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G., and Løkketangen, A. (2008). *Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography.* Springer.

Ghiani, G., Guerriero, F., Laporte, G., and Musmanno, R. (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11.

Giaglis, G. M., Minis, I., Tatarakis, A., and Zeimpekis, V. (2004). Minimizing logistics risk through real-time vehicle routing and mobile technologies: Research to date and future trends. *International Journal of Physical Distribution & Logistics Management*, 34(9):749–764.

Gillett, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349.

Glover, F. (1989). Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206.

Golden, B., Raghavan, S., and Wasil, E. (2008). *The vehicle routing problem: latest advances and new challenges*. Operations Research/Computer Science Interfaces. Springer, Dordrecht.

Gomes, C. P. and Selman, B. (1997). Algorithm portfolio design: Theory vs. practice. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 190–197. Morgan Kaufmann Publishers Inc.

Gosavi, A. (2003). *Simulation-based optimization: parametric optimization techniques and reinforcement learning*, volume 25. Springer.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128.

Grefenstette, J. J., Ramsey, C. L., and Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5(4):355–381.

Gulczynski, D., Golden, B., and Wasil, E. (2011). the multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering*, 61(3):794–804.

Haghani, A. and Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & operations research*, 32(11):2959–2986.

Haghani, A. and Yang, S. (2007). Real-time emergency response fleet deployment: Concepts, systems, simulation & case studies. In *Dynamic Fleet Management*, pages 133–162. Springer.

Hall, R. W. (2012). 2012 vehicle routing software survey. *ORMS-Today*, 39(1):408–416.

Hart, E. and Ross, P. (1998). A heuristic combination method for solving job-shop scheduling problems. In *Parallel Problem Solving from Nature*, pages 845–854. Springer.

Hartl, R. F., Hasle, G., and Janssens, G. K. (2006). Special issue on rich vehicle routing problems. *Central European Journal of Operations Research*, 14(2):103–104.

Hashimoto, H., Yagiura, M., and Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

Hentenryck, P. V. and Bent, R. (2009). *Online stochastic combinatorial optimization*. The MIT Press.

Hentenryck, P. V., Bent, R., and Upfal, E. (2010). Online stochastic optimization under time constraints. *Annals of Operations Research*, 177:151–183.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press.

Huberman, B. A., Lukose, R. M., and Hogg, T. (1997). An economics approach to hard computational problems. *Science*, 275(5296):51–54.

Hutterer, S., Vonolfen, S., and Affenzeller, M. (2013). Genetic programming enabled evolution of control policies for dynamic stochastic optimal power flow. In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion*, pages 1529–1536. ACM.

Hvattum, L. M., Løkketangen, A., and Laporte, G. (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34(4):426–438.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2006). Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40:211–225.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2007). Planned route optimization for real-time vehicle routing. In *Dynamic Fleet Management*, pages 1–18. Springer.

Jaillet, P. and Wagner, M. R. (2006). Online routing problems: Value of advanced information as improved competitive ratios. *Transportation Science*, 40(2):200–210.

Jaillet, P. and Wagner, M. R. (2008). Online vehicle routing problems: A survey. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 221–237. Springer.

Kallehauge, B., Larsen, J., Madsen, O. B., and Solomon, M. M. (2005). *Vehicle routing problem with time windows*. Springer.

Kim, K. H. and Kim, K. Y. (1999). Routing straddle carriers for the loading operation of containers using a beam search algorithm. *Computers & Industrial Engineering*, 36(1):109–136.

Kim, S., Lewis, M. E., and White III, C. C. (2005). Optimal vehicle routing with real-time traffic information. *Intelligent Transportation Systems, IEEE Transactions on*, 6(2):178–188.

Kofler, M., Beham, A., Vonolfen, S., Wagner, S., and Affenzeller, M. (2012). Modelling and optimizing storage assignment in a steel slab yard. In *Logistics and Industrial Informatics (LINDI), 2012 4th IEEE International Symposium on*, pages 101–106. IEEE.

Kofler, M., Beham, A., Wagner, S., and Affenzeller, M. (2014). Affinity based slotting in warehouses with dynamic order patterns. In *Advanced Methods and Applications in Computational Intelligence*, pages 123–143. Springer.

Kofler, M., Beham, A., Wagner, S., Affenzeller, M., and Achleitner, W. (2011). Re-warehousing vs. healing: Strategies for warehouse storage location assignment. In *Logistics and Industrial Informatics (LINDI), 2011 3rd IEEE International Symposium on*, pages 77–82. IEEE.

Kofler, M., Beham, A., Wagner, S., Affenzeller, M., and Reitinger, C. (2010). Reassigning storage locations in a warehouse to optimize the order picking process. In *22nd European Modeling and Simulation Symposium EMSS 2010, Fes, Marokko*, pages 77–82.

Kommenda, M., Kronberger, G., Wagner, S., Winkler, S., and Affenzeller, M. (2012). On the architecture and implementation of tree-based genetic programming in heuristiclab. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 101–108. ACM.

König, F. G., Lübbecke, M., Möhring, R., Schäfer, G., and Spenke, I. (2007). Solutions to real-world instances of pspace-complete stacking. In *Algorithms–ESA 2007*, pages 729–740. Springer.

Kotthoff, L. (2014). Algorithm selection literature summary. online survey. http://http://4c.ucc.ie/ larsko/assurvey/, accessed may 2014.

Kotthoff, L., Gent, I. P., and Miguel, I. (2011). A preliminary evaluation of machine learning in algorithm selection for search problems. In *Fourth Annual Symposium on Combinatorial Search*.

Koza, J. R. (1992). *Genetic Programming: vol. 1, On the programming of computers by means of natural selection*, volume 1. MIT press.

Kronberger, G. (2011). Symbolic regression for knowledge discovery: bloat, overfitting, and variable interaction networks. *ACM SIGEVOlution*, 5(4):25–25.

Kubiak, M. (2007). Distance measures and fitness-distance analysis for the capacitated vehicle routing problem. In *Metaheuristics*, pages 345–364. Springer.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.

Laporte, G. and Nobert, Y. (1983). A branch and bound algorithm for the capacitated vehicle routing problem. *Operations-Research-Spektrum*, 5(2):77–85.

Laporte, G. and Osman, I. H. (1995). Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262.

Larsen, A. (2000). *The dynamidc Vehicle Routing Problem.* PhD thesis, Technical University of Denmark, Institute of Mathematical Modelling.

Larsen, A., Madsen, O., and Solomon, M. (2002). Partially dynamic vehicle routing-models and algorithms. *Journal of the Operational Research Society*, pages 637–646.

Larsen, A., Madsen, O. B., and Solomon, M. M. (2007). Classification of dynamic vehicle routing systems. In *Dynamic Fleet Management*, pages 19–40. Springer.

Larsen, A., Madsen, O. B., and Solomon, M. M. (2008). Recent developments in dynamic vehicle routing systems. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 199–218. Springer.

Li, H. and Lim, A. (2001). A metaheuristic for the pickup and delivery problem with time windows. In *Tools with Artificial Intelligence, Proceedings of the 13th International Conference on*, pages 160 –167.

Li, J.-Q., Borenstein, D., and Mirchandani, P. B. (2007). A decision support system for the single-depot vehicle rescheduling problem. *Computers & Operations Research*, 34(4):1008–1032.

Lund, K., Madsen, O. B., and Rygaard, J. M. (1996). *Vehicle routing problems with varying degrees of dynamism.* IMM Institute of Mathematical Modelling.

Madsen, O. B., Tosti, K., and Vælds, J. (1995). A heuristic method for dispatching repair men. *Annals of operations research*, 61(1):213–226.

Mitrović-Minić, S., Krishnamurti, R., and Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669–685.

Mitrovic-Minic, S. and Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635 – 655.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Moin, N. and Salhi, S. (2007). Inventory routing problems: a logistical overview. *Journal of the Operational Research Society*, 58:1185–1194.

Moin, N. H. and Salhi, S. (2006). Inventory routing problems: a logistical overview. *Journal of the Operational Research Society*, 58(9):1185–1194.

Montemanni, R., Gambardella, L. M., Rizzoli, A. E., and Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343.

Moriarty, D. E., Schultz, A. C., and Grefenstette, J. J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276.

Nishimura, E., Imai, A., and Papadimitriou, S. (2005). Yard trailer routing at a maritime container terminal. *Transportation Research Part E: Logistics and Transportation Review*, 41(1):53–76.

Novoa, C. and Storer, R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196(2):509–515.

Oltean, M. (2005). Evolving evolutionary algorithms using linear genetic programming. *Evolutionary Computation*, 13(3):387–410.

O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., and O'Sullivan, B. (2008). Using case-based reasoning in an algorithm portfolio for constraint solving. In *Irish Conference on Artificial Intelligence and Cognitive Science*.

Pankratz, G. (2005). Dynamic vehicle routing by means of a genetic algorithm. *International Journal of Physical Distribution & Logistics Management*, 35(5):362–383.

Papastavrou, J. D. (1996). A stochastic and dynamic routing policy using branching processes with state dependent immigration. *European Journal of Operational Research*, 95(1):167–177.

Pappa, G. L., Ochoa, G., Hyde, M. R., Freitas, A. A., Woodward, J., and Swan, J. (2013). Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. *Genetic Programming and Evolvable Machines*, pages 1–33.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2012a). A review of dynamic vehicle routing problems. *European Journal of Operational Research*.

Pillac, V., Guéret, C., and Medaglia, A. L. (2012b). An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems*.

Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer.

Pitzer, E., Beham, A., and Affenzeller, M. (2014). Correlation of problem hardness and fitness landscapes in the quadratic assignment problem. In *Advanced Methods and Applications in Computational Intelligence*, pages 165–195. Springer.

Pitzer, E., Beham, A., Affenzeller, M., Heiss, H., and Vorderwinkler, M. (2011). Production fine planning using a solution archive of priority rules. In *Logistics and Industrial Informatics (LINDI), 2011 3rd IEEE International Symposium on*, pages 111–116. IEEE.

Pitzer, E., Vonolfen, S., Beham, A., Affenzeller, M., Bolshakov, V., and Merkuryeva, G. (2012). Structural analysis of vehicle routing problems using general fitness landscape analysis and problem specific measures. In *14th International Asia Pacific Conference on Computer Aided System Theory*, pages 36–38.

Polacek, M., Hartl, R. F., Doerner, K., and Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, 10(6):613–627.

Potvin, J.-Y. and Bengio, S. (1996). The vehicle routing problem with time windows part ii: genetic search. *INFORMS journal on Computing*, 8(2):165–172.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

Powell, W. B. (2012). Perspectives of approximate dynamic programming. *Annals of Operations Research*, pages 1–38.

Powell, W. B., Sheffi, Y., Nickerson, K. S., Butterbaugh, K., and Atherton, S. (1988). Maximizing profits for north american van lines' truckload division: A new framework for pricing and operations. *Interfaces*, 18(1):21–41.

Powell, W. B., Simao, H. P., and Bouzaiene-Ayari, B. (2012). Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal on Transportation and Logistics*, 1(3):237–284.

Psaraftis, H. (1988). Dynamic vehicle routing problems. In *Vehicle Routing: Methods and Studies*, pages 223–249. Elsevier Science Publishers.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *annals of Operations Research*, 61(1):143–164.

Pureza, V. and Laporte, G. (2008). Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. *INFOR: Information Systems and Operational Research*, 46(3):165–176.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65–118.

Richter, A. (2005). *Dynamische Tourenplanung: Modifikation von klassischen Heuristiken für das Dynamische Rundreiseproblem (DTSP) und das Dynamische Tourenplanungsproblem (DVRP) mit der Möglichkeit der Änderung des aktuellen Fahrzeugzuges*. PhD thesis, TU Dresden.

Richter, H. (2009). Detecting change in dynamic fitness landscapes. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 1613–1620. IEEE.

Ropke, S. (2012). Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. In *International Workshop on Column Generation 2012*, Bromont, Quebec, Canada.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4):258–272.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Ruiz-Vanoye, J. A., Díaz-Parra, O., Landero, V., et al. (2008). Applied statistical indicators to the vehicle routing problem with time windows for discriminate appropriately the best algorithm. In *Computational Science and Its Applications–ICCSA 2008*, pages 1131–1141. Springer.

Ruland, K. and Rodin, E. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & mathematics with applications*, 33(12):1–13.

Scheibenpflug, A., Wagner, S., Kronberger, G., and Affenzeller, M. (2012). Heuristiclab hive-an open source environment for parallel and distributed execution of heuristic optimization algorithms. In *1st Australian Conference on the Applications of Systems Engineering ACASE'12*, page 63.

Schmid, V., Doerner, K. F., Hartl, R. F., and Salazar-González, J.-J. (2010). Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Computers & Operations Research*, 37(3):559–574.

Schmid, V., Doerner, K. F., Hartl, R. F., Savelsbergh, M. W., and Stoecher, W. (2009). A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1):70–85.

Schwefel, H. (1977). Numerische optimierung von computer-modellen. interdisciplinary systems research, vol. 26.

Silverthorn, B. C. (2012). *A Probabilistic Architecture for Algorithm Portfolios*. PhD thesis, The University of Texas at Austin. May.

Sleator, D. D. and Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208.

Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Swihart, M. R. and Papastavrou, J. D. (1999). A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *European Journal of Operational Research*, 114(3):447–464.

Taillard, É. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186.

Tang, L., Liu, J., Rong, A., and Yang, Z. (2001a). A review of planning and scheduling systems and methods for integrated steel production. *European Journal of Operational Research*, 133(1):1–20.

Tang, L., Liu, J., Rong, A., and Yang, Z. (2002). Modelling and a genetic algorithm solution for the slab stack shuffling problem when implementing steel rolling schedules. *International Journal of Production Research*, 40(7):1583–1595.

Tang, L., Liu, J., Rong, A., Yang, Z., et al. (2001b). An effective heuristic algorithm to minimise stack shuffles in selecting steel slabs from the slab yard for heating and rolling. *Journal of the Operational Research Society*, 52(10):1091–1097.

Tang, L. and Ren, H. (2010). Modelling and a segmented dynamic programming-based heuristic approach for the slab stack shuffling problem. *Computers & Operations Research*, 37(2):368–375.

Thompson, P. M. and Psaraftis, H. N. (1993). Cyclic transfer algorithm for multivehicle routing and scheduling problems. *Operations research*, 41(5):935–946.

Tompkins, J. A. (2010). *Facilities planning*. John Wiley & Sons.

Toth, P. and Vigo, D. (2002). *The vehicle routing problem*, volume 9. Siam.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346.

van Hemert, J. I. and La Poutré, J. A. (2004). Dynamic routing problems with fruitful regions: Models and evolutionary computation. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 692–701. Springer.

van Lon, R. R., Holvoet, T., Vanden Berghe, G., Wenseleers, T., and Branke, J. (2012). Evolutionary synthesis of multi-agent systems for dynamic dial-a-ride problems. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 331–336. ACM.

Ventresca, M., Ombuki-Berman, B., and Runka, A. (2013). Predicting genetic algorithm performance on the vehicle routing problem using information theoretic landscape measures. In *Evolutionary Computation in Combinatorial Optimization*, pages 214–225. Springer.

Vidal, T., Gabriel Crainic, T., Gendreau, M., and Prins, C. (2012). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*.

Vonolfen, S., Affenzeller, M., Beham, A., Lengauer, E., and Wagner, S. (2013a). Simulation-based evolution of resupply and routing policies in rich vendor-managed inventory scenarios. *Central European journal of operations research*, 21(2):379–400.

Vonolfen, S., Affenzeller, M., Beham, A., Wagner, S., and Lengauer, E. (2011a). Simulation-based evolution of municipal glass-waste collection strategies utilizing electric trucks. In *Logistics and Industrial Informatics (LINDI), 2011 3rd IEEE International Symposium on*, pages 177–182. IEEE.

Vonolfen, S., Affenzeller, M., and Wagner, S. (2012a). Modeling rich and dynamic vehicle routing problems in heuristiclab. In Bruzzone, A., Gronalt, M., Merkuryev, Y., Piera, M., and Talley, W., editors, *Proceedings of the 14th International Converence on Harbor, Maritime and Multimodal Logistics Modelling and Simulation (HMS 2012)*, pages 96–102. DIPTEM University of Genova.

Vonolfen, S., Beham, A., Affenzeller, M., Wagner, S., and Mayr, A. (2011b). Combination and comparison of different genetic encodings for the vehicle routing problem. In *Proceedings of the 13th international conference on Computer Aided Systems Theory-Volume Part I*, pages 327–334. Springer-Verlag.

Vonolfen, S., Beham, A., Kofler, M., Affenzeller, M., and Doerner, K. (2013b). Simulation-based optimization of transport activities within cold charge steel production. In *Logistics and Industrial Informatics (LINDI), 2013 5th IEEE International Symposium on*. IEEE.

Vonolfen, S., Beham, A., Kommenda, M., and Affenzeller, M. (2013c). Structural synthesis of dispatching rules for dynamic dial-a-ride problems. In *Proceedings of the 14th international conference on Computer Aided Systems Theory*. Springer-Verlag.

Vonolfen, S., Kofler, M., Beham, A., Affenzeller, M., and Achleitner, W. (2012b). Optimizing assembly line supply by integrating warehouse picking and forklift routing using simulation. In *Proceedings of the Winter Simulation Conference*, page 339. Winter Simulation Conference.

Vonolfen, S., Wagner, S., Beham, A., Kofler, M., Affenzeller, M., Lengauer, E., and Scheucher, M. (2010). A simulation-based approach to the vehicle routing problem. In *22nd European Modeling and Simulation Symposium EMSS 2010, Fes, Marokko*, pages 363–368.

Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., Vonolfen, S., Kofler, M., Winkler, S., Dorfer, V., and Affenzeller, M. (2014). Architecture and design of the heuristiclab optimization environment. In *Advanced Methods and Applications in Computational Intelligence*, pages 197–261. Springer.

Waller, M., Johnson, M., and Davis, T. (1999). Vendor-management inventory in the retail supply chain. *Journal of Business Logistics*, 20:181–203.

Wark, P. and Holt, J. (1994). A repeated matching heuristic for the vehicle routeing problem. *Journal of the Operational Research Society*, pages 1156–1167.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Whiteson, S. (2012). Evolutionary computation for reinforcement learning. In *Reinforcement Learning*, pages 325–355. Springer.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82.

Xu, L., Hoos, H., and Leyton-Brown, K. (2010). Hydra: Automatically configuring algorithms for portfolio-based selection. In *AAAI*, volume 10, pages 210–216.

Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2008). Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Intell. Res.(JAIR)*, 32:565–606.

Yang, J., Jaillet, P., and Mahmassani, H. (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38(2):135–148.

Zaepfel, G. and Vogl, P. (2010). *Dynamisch-adaptive Tourenplanung*. Trauner Verlag.

Zak, J. (2010). Decision support systems in transportation. In *Handbook on Decision Making*, pages 249–294. Springer.

Zäpfel, G. and Wasner, M. (2006). Warehouse sequencing in the steel supply chain as a generalized job shop model. *International Journal of Production Economics*, 104(2):482–501.

Zeimpekis, V., Minis, I., Mamassis, K., and Giaglis, G. (2007). Dynamic management of a delayed delivery vehicle in a city logistics environment. In *Dynamic Fleet Management*, pages 197–217. Springer.

Curriculum Vitae

Personal information

Surname(s) / First name(s)	**Stefan Vonolfen, MSc**
Address(es)	Kornrödt 3, 4742 Pram, Austria
Email(s)	stefan.vonolfen@gmail.com
Date of birth	June 30, 1986
Gender	Male

Work experience

Dates	**January 2010 - May 2014**
Occupation or position held	Research and teaching associate
Name and address of employer	Upper Austria University of Applied Sciences, research group HEAL (http://heal.heuristiclab.com)
Type of business or sector	Research and development
Dates	**January 2009 - July 2009**
Occupation or position held	Software engineer (part-time)
Name and address of employer	Ubitronix System Solutions GmbH, Hagenberg (Austria)
Type of business or sector	Energy management solutions
Dates	**April 2008 - September 2008**
Occupation or position held	Software engineer (internship)
Name and address of employer	Siemens Corporate Research, Princeton NJ (USA)
Type of business or sector	Automation and drives
Dates	**August 2006 - September 2006**
Occupation or position held	Summer internship
Name and address of employer	RZL Software, Ried im Innkreis (Austria)
Type of business or sector	Accounting software

Education and training

Dates	**Since October 2010**
Course name	Doctoral Programme in Technical Sciences / Informatics
Name of organization	University of Linz (Austria)
Thesis topic	Adaptive Heuristic Approaches for Dynamic Vehicle Routing - Algorithmic and Practical Aspects
Main subjects	Operations research, simulation, metaheuristics

Dates	**March 2012**
Course name	PhD course on Local Distribution Planning
Name of organization	Molde University College (Norway)
Main subjects	Heuristic and exact algorithms for vehicle routing
Dates	**October 2008 - July 2010**
Title of qualification awarded	Master of Science in Engineering (with distinction)
Name of organization	University of Applied Sciences Hagenberg (Austria)
Thesis topic	A Domain-Specific Language for Heuristic Optimization
Main subjects	Software engineering
Dates	**July 2009 - November 2009**
Course name	Study abroad semester
Name of organization	Edith Cowan University in Perth (Australia)
Main subjects	Computer science
Dates	**October 2005 - September 2008**
Title of qualification awarded	Bachelor of Science in Engineering (with distinction)
Name of organization	University of Applied Sciences Hagenberg (Austria)
Thesis topic	Evolving Objects – A Framework for Evolutionary Computation
Main subjects	Software engineering

Additional information

Guest Lectures

- Guest lecture on Heuristic Optimization at Technical University Riga (Latvia, October 2013)

Awards

- Multiple scholarships for excellent merits in studies from the University of Applied Sciences Upper Austria for the years 2005-2010
- Best thesis award from the University of Applied Sciences Upper Austria in the year 2010

Voluntary Work

- Work as a paramedic at the Red Cross (2005 - 2010)