Vehicle Routing under Uncertainty

Tamás Máhr

Vehicle Routing under Uncertainty

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 21 september 2011 om 10:00 uur

door
Tamás MÁHR,
Master of Science in Technical Informatics,
Budapest University of Technology and Economics
geboren te Boedapest, Hongarije.

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. C. Witteveen

Copromotor: Dr. M.M. de Weerdt

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. C. Witteveen	Technische Universiteit Delft, promotor
Dr. M.M. de Weerdt	Technische Universiteit Delft, copromotor
Prof. dr. P. Davidsson	Blekinge Institute of Technology
Prof. dr. ir. J.A. La Poutré	Technische Universiteit Eindhoven / CWI
Prof. dr. R.A. Zuidwijk	Technische Universiteit Delft / Erasmus University Rotterdam
Prof. dr. R. Babuska	Technische Universiteit Delft
Dr. A.H. Salden	Almende BV
Prof. dr. ir. G.J.P.M. Houben	Technische Universiteit Delft, reservelid

This work was carried out partly at Almende BV, and partly at the Algorithms group of the Software Engineering department of the Delft University of Technology. From 1st April 2004 till 1st April 2006, the PhD research was performed as part of the Distributed Engine for Advanced Logistics project (DEAL; EETK01141) that was carried out within the Programme E.E.T. and subsidised by Senternovem.

TRAIL Thesis Series T2011/11, the Netherlands TRAIL Research School

TRAIL Research School
PO Box 5017
2600 GA Delft
The Netherlands
T: +31 (0) 15 278 6046
F: +31 (0) 15 278 4333
E: info@rsTRAIL.nl

Cover Design: Orsolya Árva

ISBN 978-90-5584-148-6

Printed in Hungary.

Contents

Chapter 1

Introduction

1

Cooperation in a group (of people, companies, etc.) emerges when the participants realize that they are more efficient working together, and they are willing to compromise on their individual preferences. If they can act as a group, and organize their operations collectively, it is usually possible to achieve substantial savings. In modern times, probably the first recorded example of operations research is that of Charles Babbage (1791-1871) (Sodhi, 2007). He is, of course, best known for designing the first mechanical computing machine, but he also pioneered the field of cost calculations for transportation and sorting mail. Ultimately, his work on organising the mail led to the introduction of England's universal "Penny Post" in 1840.

To elaborate on the benefits of organizing operations, we should look at how the scientific field *operations research* emerged. In World War II, British and American armed forces sought ways to make better decisions in organizing military operations. A team of researchers, led by P.M.S. Blackett in the Admirality's Operational Research unit, discovered for example that defense of larger transatlantic convoys is more efficient than that of small convoys. "A convoy of sixty merchant ships and twelve escorts was in fact much more strongly protected than a convoy of thirty ships and six escorts." (Fisher, 2000). They described convoy battles as *unstable equilibrium* where, when advantage went to one side, it went completely to that side. Indeed, when escort forces were regrouped to double the size of defensive forces over the North Atlantic in April and May 1943, the balance turned over decisively in favor of the defenders.

Since the war, operations research (sometimes also called as *management science*) has also found uses in civil applications, of which some typical examples include:

- designing the layout of a factory for the efficient flow of materials,
- constructing a telecommunications network at low cost while still guaranteeing quality of service (QoS),
- determining the routes of school buses (or city buses) to minimize the number of buses needed,
- managing the flow of raw materials and products in a supply chain based on uncertain demand for the finished products,
- managing freight transportation and delivery systems, and
- scheduling jobs on a number of processors.

Most of these applications are related to transportation of people or goods within a location or between locations. Our current focus is on freight transportation.

The importance of efficient organization in freight transportation is best shown by the 2007 report of the European Commission on energy and transport (European Commission, Directorate-General for Energy and Transport and Eurostat, 2007). In 2006, the transport services sector (not only freight) employed about 8.8 million persons in the 27 countries of the EU. The demand for land-based freight transport (road, rail, inland waterways and pipelines) added up to 2,595 million tonne-kilometer, which grew by 2.8% per year on average in 1995-2006. The freight transportation over the road alone was estimated to be approximately 258 billion euros. With such numbers, representing a big impact on the economy, efficient organization of (freight) transportation can achieve substantial savings. On the micro level, the competition among transportation companies is fierce, therefore failing to use the resources efficiently could cost a company its survival.

In most transportation companies, organization of the operations is performed by humans. A European transportation company in the Netherlands that owns approximately 3000 trucks needs more than 200 human planners to coordinate transportation. The planners spend two to three years learning the operational details and all the relevant constraints. The coordination of so many trucks requires tight cooperation among the human planners. Their work has these phases. First they have to generate plans for the trucks, then they have to monitor whether the trucks can follow these plans, and finally they have to modify the plans to accommodate changes.

The field of operations research offers assistance to human planners in all of these tasks. The algorithms mainly focus on planning, but continuous re-planning for adaptation to changes is also supported to some extent. In the original setting, an OR planner is used to calculate optimal plans for the next day, and during execution, human planners adjust the plans by hand to react to changes.

Recent advancements in technology, however, are changing this practice. Computer systems controlling and monitoring factories can provide information about the production state of goods, which can be used to plan the transportation of those goods more efficiently. Other sources of information include *Geographic Information Systems* (GIS) in connection with the *Global Positioning System* (GPS). Such tools are capable of continuously monitoring the state and position of vehicles, thus providing real-time information about the actual state of transportation. These new technologies provide an overwhelming amount of important information to the human planners, who can exploit computerized tools to make planning decisions on-the-fly.

The demand for continuous planning methods propelled research on *dynamic transportation problems*, on problems where things change, and where algorithms have a chance to react to those changes. In order to to specify our research questions related to these problems, the following section first states the difficulty of dynamic transportation planning, and then introduces centralized and distributed approaches for such problems. Section 1.2 lists some open issues regarding the challenge of being efficient and robust. Section 1.3 shows how this thesis contributes to the solution of these challenges. Finally, the chapter ends by providing a quick outline of the thesis in Section 1.4.

1.1 Approaches to Dynamic Transportation Problems

Using up-to-date GIS technology, transportation companies can monitor the activities of their vehicles, and modify their operations, whenever necessary. Such necessities arise when the execution of plans is disturbed (by traffic jams, or truck breakdowns for example), or when new trucks or new jobs arrive. Some of these events (the execution incidents) are sensed by the vehicles or their drivers and communicated to the planners, while others (new trucks or orders) reveal themselves immediately to the planners. One could categorize such events as events that introduce new elements to the problem like a new order, a new truck, or a failing truck. Other events do not introduce new elements, but they change existing parameters. A congestion event changes the travel times between locations, loading/unloading delays change the service times required by orders.

Receiving an almost continuous flow of information about the actual state of their trucks, companies need to find a way to use this information to improve decisions made during execution. The methods and solutions provided by the operations research field can be used to compute efficient plans using this information. These methods, which are centralized by nature, can use every piece of relevant information that is available for computing the plans. Any unpredicted change in the state of the world, however, invalidates the previously computed plans. Most methods then require a complete re-run to produce new plans for the new situation, but there are also some continuous heuristics (based on e.g. genetic algorithms – see Section 2.2.1) that are capable of incorporating changes without requiring a restart. Even so, all of these methods consider all available information, and thereby require a consistent global state. This requirement can be satisfied as long as the changes are infrequent. There needs to be a period of time during which we can assume a static state of the world. As modern information communication technologies develop, this assumption holds less and less.

When changes in different parts of the modeled problem occur frequently, one may not assume having a consistent picture of the actual state of the affairs. Updates from different vehicles suffer different delays, therefore at any time point, a central entity may have received only a part of the updates. Information local to certain trucks may be consistent, but the overall picture may not. This suggests to approach the problem in such a way that assumes local consistency only. One could, for example, represent all vehicles and orders as individual entities, and provide these representative entities some means of communication. Then formulate goals for every such communicating entity, and equip them with some decision procedures that they can use together with their communication protocols to reach their goals. If the decision procedures rely only on local information encapsulated by any singe entity, plus the information it may receive via communicating with other entities, then it is enough to ensure consistency of this information. If any decision involves information from a small set of the entities only, this consistency is much easier to ensure than global consistency.

The distributed decision making entities described above are called *agents*, and the group of agents and their communication protocols together are called *Multi-agent Systems* (MAS). A multi-agent approach to dynamic transportation problems has the following properties:

- Since information about one truck is gathered by the truck itself, and sent to the planning system (containing the agents) in one package, truck agents maintain a consistent state,

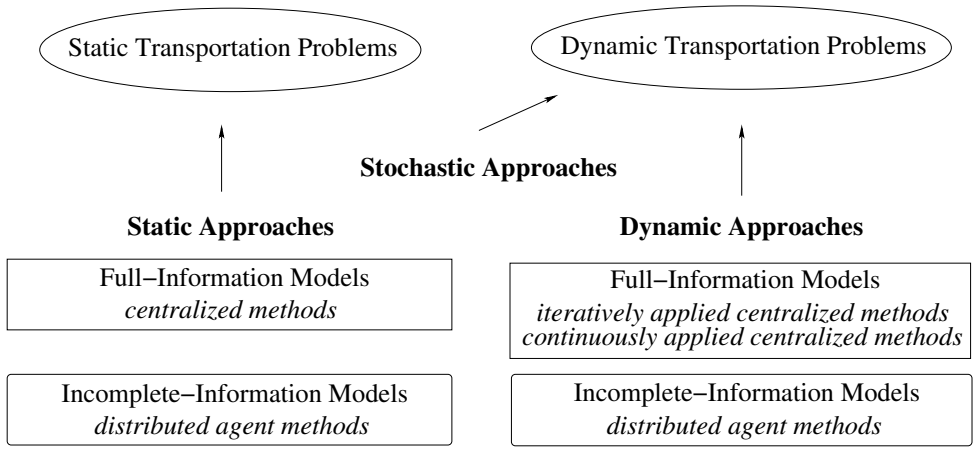


Figure 1.1: The relation of transportation problems and the possible approaches.

even though the combined state of all agents might be inconsistent. Similar reasoning holds for other entities too.

- Agents make local changes only, therefore problems introduced by the inconsistent global state are kept local. A centralized algorithm would reorganize the whole solution to accommodate a new state that is assumed to be consistent.

- Agents solve problems by communicating their (local) states to each other. Whenever such a state changes due to an event, the corresponding agent will simply continue by communicating the new state. There is no need to restart the whole process for every change.

Multi-agent models as described above are *incomplete-information models*, because individual agents consider only part of the relevant information that is available to the system as a whole. In contrast, classical centralized models are *full-information models*, because they do use all available information. In cases where there is a consistent global state information available (for example in *static transportation problems*, where nothing changes after the problem is given), incomplete-information models impose unnecessary constraints, and therefore their application is less common. However, they are frequently applied in *dynamic transportation problems*, where the problem changes from time to time.

One alternative to solve dynamic transportation problems is to use *stochastic approaches* (see Figure 1.1), where we assume some probabilistic information about future changes (see Powell et al., 1995). Using this information, it is possible to compute transportation plans that are optimal in the expectation of the probabilistic events. If proper stochastic information is lacking, *dynamic approaches* can be applied that continuously monitor the world, observe the changes, and react to them. These approaches produce plans ignoring the possible changes, but adjust the plans in reaction to every change. Dynamic approaches may use either classical centralized, full-information, or agent-based, incomplete-information models.

In real life, when transportation plans are executed, various things can go wrong. Trucks break down, traffic jams hinder the vehicles, loading and unloading might take more time than expected. These events do not have a direct effect on the planners, but on the vehicles executing the plans. Accordingly, algorithms applied on dynamic problems need to be compared by

evaluating the *execution* instead of the plans. When dynamic events are present, the execution can be very different from what was planned.

By comparing the execution of plans that resulted from different planning methods for problems with dynamic events, it is important to know which method is more *robust* against certain events. The *Free On-line Dictionary of Computing* gives the following definition of the word 'robust':

> "Said of a system that has demonstrated an ability to recover gracefully from the whole range of exceptional inputs and situations in a given environment. One step below bulletproof. Carries the additional connotation of elegance in addition to just careful attention to detail. Compare *smart*, opposite: *brittle*."

To interpret this in the context of logistical planning, we call a planning method robust, if it is minimally disturbed by unforeseen events. It may be disturbed, but we want this disturbance to be as small as possible. In our opinion, it is interesting to quantify this disturbance and use it as a point of comparison when evaluating a planning method on quality. Such a robustness vs. quality comparison can provide useful insights to the dynamic behavior of planning methods.

1.2 Research Questions

The two main themes of dynamism in vehicle routing problems and the evaluation of planning methods for dynamic vehicle routing are encapsulated in the research questions this thesis addresses. In the following, we formulate research questions in both topics. First, we formulate a high-level and two more concrete questions about dynamism in vehicle routing problems. Then we formulate another research question related to the comparison of centralized optimization and agent-based planning methods.

Previous studies of dynamic transportation problems are mainly restricted to problems with new order arrivals. There is some literature on transportation with travel times that depend on the time of day, but a comprehensive framework for describing dynamic events as sources of uncertainty in transportation problems and quantifying the effects of these uncertainties is still missing. Therefore, we formulate the following research question that asks for a formal description of possible sources of uncertainty in vehicle routing, and suggests the development of a framework for analyzing such problems.

Research Question 1 *How should uncertainties in transportation problems be formalized, and what kind of framework should be developed to analyze such problems?*

Vehicle-routing problems are formalized by defining the input problem, the solution of the problem planning methods are expected to output, and the measure that is computed from the solution in order to assess the quality of the planning method. In a similar fashion, when formalizing uncertainties in vehicle routing, one should define the uncertainties in the input, the solution to the extended problem , and the measure to judge the method that handled the uncertainties. While the same measure could be used in problems with and without uncertainties to judge the performance of the method, this may not be sufficient to assess the effect of uncertainties on the method. Therefore, the following research question raises the issue of quantifying the effect uncertainties have on planning methods.

Research Question 1a *How can we quantify the effects of uncertainties on planning methods?*

Finally, since we would like to compare different planning methods in problem instances with different uncertainties, it is important to think about the proper way to carry out such experiments. The traditional way of measuring the plans produced by the planning methods does not seem to be sufficient here, since it does not properly express the effect of dynamic events. Therefore, we formulate the following question to study this problem.

Research Question 1b *What are the requirements of a simulation test-bed that is used to evaluate planning methods in transportation problems with uncertainty?*

In addition to studying the effect of uncertainties on planning methods, we are also, (and possibly even more) interested in the other dimension of the problem. Approaching the problem from the perspective of the planning methods, we would like to know whether distributed agent-based planning methods, or centralized optimization-based planning methods perform better under uncertainty. This problem is captured in the following research question.

Research Question 2 *How do uncertainties in transportation problems affect centralized and distributed solution methods of such problems?*

The research carried out in pursuit of the above questions yielded four contributions to the state of the art in vehicle routing. These contributions are the topic of the next section.

1.3 Contributions

Driven by the research questions presented in the previous section, this thesis documents a study of vehicle routing problems under uncertainty from two different points of view. From the point of view of uncertainties, the effect of different types of uncertainties on planning methods are analyzed. From the point of view of planning methods, the performance of agent-based and centralized on-line optimization-based planning methods in uncertain problem instances are compared.

The substitution heuristic. In Chapter 4, we presented a new lightweight heuristic (the substitution heuristic) that appears superior to existing heuristics for distributed transportation problems. This new heuristic is an extension of the insertion heuristic. It is motivated by the *decommitment* idea of leveled-commitment contracts, where breaking a contract is allowed if the value of a new contract is greater than the contractor's commitment to the current contract (i.e. the amount of penalty the contractor must pay). In a similar fashion, if a newly inserted job fits the plan better, the new heuristic allows the substitution of an already planned job in the plan of a vehicle. Instead of a penalty, the substitution depends on the market position of the removed order, thereby avoiding the removal of orders that would have difficulties finding another truck. This local decision of the truck results in overall better plans than simple adaptation of classical improvement methods to distributed problems. Due to its distributed nature, our heuristic is more appropriate in real environments, where global decision making is not feasible by neither centralized optimization nor classical heuristics.

Agent-based heuristics vs. on-line optimization. One of the main questions of this research (Research Question 2) is whether a distributed computationally lightweight approach (the agent approach) could be an alternative for classical centralized optimization approaches in vehicle routing problems with uncertainty. For each uncertainty type included in out experiments (new order arrivals, trucks breakdowns, and service-time variations), we identified the critical uncertainty level, where the performance of the agent-based solutions and the centralized on-line optimization are equal, and which separates domains where one approach dominates the other. We found that some sources of uncertainty (new order arrivals, and service-time variations) were handled by the agent methods better, but we also found one kind of uncertainty (truck breakdowns) that favoured the centralized optimization method. In conclusion, we can say that our experimental research showed that lightweight distributed local heuristics are competitive with centralized optimization methods, and in particular show the same performance profiles (or sometimes even better) in transportation problems with uncertainty.

Beside the above two main contributions we further list two more that can be seen as part of the comparison study of distributed heuristics and centralized optimization methods, but which are the results of our attempt to answer Research Questions 1, 1a, and 1b.

A model of transportation problems with uncertainties. In order to systematically study the effects of environmental disturbances on the performance of transportation-planning methods, this thesis extends classical transportation-problem formulations by a framework of uncertainties in Chapter 3. The framework allows for the definition of different sources of uncertainties that influence plan execution independently of the static transportation problem at hand. Any previously defined static transportation problem can be turned into a dynamic transportation problem using this framework. By separating the definition of the static transportation problem and the sources of uncertainties, the framework facilitates the reuse of transportation problems and incident definitions. Related to the uncertainties introduced, the framework defines an alternative performance measure to quantify the robustness of transportation-planning methods. Combined, these contributions address Research Questions 1 and 1a.

Simulation framework. To answer Research Question 2 on whether a lightweight distributed approach can be an alternative to centralized optimization in transportation problems with uncertainties, we conducted a series of computational experiments with the help of a simulation framework developed for this research. To be able to perform the simulations we had to answer Research Question 1b about the requirements of a simulation test bed. We found that a real-time simulation environment with a separation of planning for and simulation of vehicle movements best achieved the analytical needs of transportation problems with uncertainties. In accordance with these findings, we developed a distributed simulation environment, as described in Chapter 5, to compare our distributed and centralized transportation-planning methods.

After summarizing our major and minor contributions, hereby we provide a list of publications related to this thesis. To start with, the software architecture of the underlying agent system, as described in Chapter 4, was first discussed in a 2004 paper presented at the BNAIC conference.

> Máhr, T. and de Weerdt, M. M. (2004). Distributed agent platform for advanced logistics. In *Proceedings of the Belgium-Dutch Conference on Artificial Intelligence*

(BNAIC-04), pages 395–396. BNVKI.

The first comparison of the centralized optimization method and the agent-based heuristics (as presented in Chapter 4) in a dynamic drayage problem was discussed at the ATT workshop of the AAMAS in 2008. This paper introduces the real-world container transportation problem with dynamic release times, and presents the results of simulations with the two solution methods, both of which are included in Chapter 6 in this thesis.

> Máhr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2008). Agent performance in vehicle routing when the only thing certain is uncertainty. In *Proceedings of the workshop on Agents in Traffic and Transportation (ATT)*.

This work was extended to include service-time uncertainty scenarios and a more detailed description of the agent approach in a journal paper published in Transportation Research Part C in 2010. This paper pioneers in studying the joint effect of release-time uncertainty and service-time uncertainty, and as such it contributes mainly to Chapter 6.

> Máhr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2010). Can agents measure up? a comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research Part C: Emerging Technologies*, 18(1):99 – 119. ISSN 0968-090X. Information/Communication Technologies and Travel Behaviour; Agents in Traffic and Transportation.

To support other researchers interested in dynamic transportation problems, the release-time uncertainty scenarios that were derived from a real-world problem in Chapter 6 were submitted to the MIPLIB[1], „an electronically available library of both pure and mixed integer programs". This submission was reported in the ERIM report series in 2010.

> Srour, F. J., Máhr, T., de Weerdt, M. M., and Zuidwijk, R. A. (2010). MIPLIB truckload PDPTW instances derived from a real-world drayage case. In *ERIM report series research in management Erasmus Research Institute of Management*. Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam (ERIM is the joint researchinstitute of the Rotterdam School of Management, Erasmus University and the Erasmus School of Economics (ESE) at Erasmus University Rotterdam).

Finally, the list of studied uncertainties were extended with truck-breakdown uncertainty (from Chapter 6) in a conference paper in 2011. This paper also introduced our robustness measure from Chapter 3, and discussed the major properties of the simulation framework as presented in Chapter 5

> Máhr, T., Srour, J., and de Weerdt, M. M. (2011). Using simulation to evaluate how multi-agent transportation planners cope with truck breakdowns. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control*. IEEE.

[1] `http://miplib.zib.de/`

To summarize, via the papers listed above, this thesis contributes to the state of the art in vehicle routing, by i) introducing a new distributed algorithm (the substitution method) for vehicle routing, ii) providing experimental results on comparing agent-based heuristics and centralized on-line optimization for vehicle routing under uncertainty , iii) providing a framework to extend existing vehicle-routing problems with uncertainties, and iv) describing a simulation environment designed to evaluate planning methods in vehicle-routing problems with uncertainty. The next and final section of this chapter explains the structure of the thesis.

1.4 Organization of the document

In this thesis, we compare two different transportation planning methods using a vehicle routing problem with different kinds of uncertainties. In order to do this, we need three things. We need a problem definition with uncertainties, we need different planning methods that can solve this problem, and we need to define how exactly we will compare the performance of the methods. The structure of the thesis roughly follows this logic.

First, Chapter 2 describes what models and solution methods have already been developed by other studies for different vehicle routing problems. The chapter is divided into two parts. The first part describes problem definitions tailored to centralized and distributed approaches. The second part surveys the respective solution methods that were successfully applied to vehicle routing problems. The chapter ends with a short overview of the performance indicators that real-world transportation companies use to evaluate their own performance.

After this overview of the literature, the thesis sets out by formally defining a vehicle routing problem with uncertainties in Chapter 3. Three different kinds of uncertainties are defined that can be part of an actual problem instance individually, or in combination. Subsequently, the problem is reformulated in two different ways, one that fits centralized optimization algorithms, and another one that fits distributed agent-based solution methods. Finally, this chapter formally defines robustness as a measure of logistical performance. This measure will be used to compare different planning methods for the vehicle routing problems defined in this chapter.

The next chapter, Chapter 4, describes the different planning methods that are compared in the thesis. First, an agent-based solution method is introduced. This method is a collection of three distributed algorithms, adopted from the literature, and improved for better performance. Then a centralized on-line optimization method is introduced, which is also an adaptation of an existing technique. Finally, a short section analyses the key differences between the two approaches.

With the vehicle routing problem formalized and the different methods introduced, Chapter 5 explains how these methods are compared. We present a distributed real-time simulation environment designed to evaluate vehicle routing solutions under uncertainty. The chapter also describes the software architecture of the two solution methods, as well as the simulator, to provide insight on the implementation.

After introducing all the ingredients of the comparison study, Chapter 6 describes the experimental design. First, a real-world transportation case is described as the basis of the scenarios used in the comparison study, and then the concrete parameter settings are given. The discussion of the results starts with the evaluation of the different algorithms of the agent-based

approach. Then the behavior of the agent-based and the centralized on-line optimization methods are compared on problems with single and multiple sources of uncertainty. Finally, the effects that the different uncertainties have on the methods are compared.

The final chapter, Chapter 7, summarizes the contributions of this thesis by answering the research questions from this chapter. In the last section of this book, some promising directions for future research are given.

Chapter 2

Transportation Problems and Solution Approaches

2

Planning and execution of transport are crucial but complex activities in various applications. Military logistics (Perugini et al., 2003b,a) as well as public applications (Fischer et al., 1995; van der Putten et al., 2006) suffer the complexity of planning and the difficulty of executing transportation in an ever-changing environment. Computing plans can, in the first place, be difficult. If the plans need to be recomputed every time something changes, that further complicates matters. Due to the importance of logistic applications, serious effort has been put into researching the basic problems of transportation planning, as well as the problems introduced by new information. The main goal has been to produce algorithmic tools to help humans solve these problems.

In order to develop such tools, researchers have identified several different abstractions within the field of transportation. These include the *Vehicle-Routing Problem* (Dantzig and Ramser, 1959), the *Pickup and Delivery Problem* (Savelsbergh and Sol, 1995), or the *Dial-a-Ride Problem* (Psaraftis, 1980, 1983), and variations of these. These problems share a common point of view: they all focus on the logistics company, the company that owns vehicles and accepts orders. Their goal is to find efficient routes for the trucks to service all orders. Routes are selected to minimize an objective function, which emphasizes the company's interest (overall costs). An underlying assumption of this company-oriented view is that all information about the problem is available to a central decision maker in the company.

In addition to such a centralized company-oriented view of transportation problems, these problems can also be seen as *Task Allocation Problems* (Sandholm, 1999). This problem is commonly studied in connection with *Multi-Agent Systems*. Multi-agent models constructed for the task allocation problem concentrate on distributed decision making entities (the agents), and their negotiation processes. Such models focus on local preferences and constraints, rather than global goals. An important property of such problem formulations is that full-information is not required by any individual part of the system.

The rest of this chapter is dedicated to the literature of the two problem families. First, problem definitions of both the classical operations research (OR) vehicle-routing problems and the multi-agent, distributed task-allocation problem are given in Section 2.1. Then, general exact optimization methods and special heuristics for vehicle-routing problems are discussed along with market mechanisms for multi-agent methods in Section 2.2. After this overview of centralized and distributed problem formulations and solution approaches, a short overview of key performance indicators used by real transportation companies is given in Section 2.3. Finally, the chapter concludes with a summary of the gaps identified in the literature.

2.1 Problems

The common view of the transportation problems, formulated in the field of operations research, is that there is a company with some trucks and orders; the goal is to find routes for the trucks such that every order is included in exactly one route, in such a way that a predefined cost function is minimized. This general form defines a global point of view, where the emphasis is on the company's objectives. *All* information, known to the system, is assumed to be available for a central algorithm to make decisions.

In contrast, multi-agent solutions to transportation problems, usually based on a task-allocation formulation, take a local point of view where the knowledge of all information is not assumed. Different players in the problem are modeled as agents, each of which makes its decisions independently and autonomously of the others. In such formulations, multiple contradicting interests can be modeled as different agents, and the decisions of agents need only the limited information that is available to the individual agent.

The following subsections describe these two problem formulations in greater detail. First Section 2.1.1 introduces the literature of vehicle-routing problems, and then Section 2.1.2 that of task-allocation problems.

2.1.1 Vehicle-Routing Problems

The most basic vehicle-routing problem is the *Traveling Salesman Problem* (TSP). For a detailed discussion about the central importance of this problem, see (Lawler et al., 1985). The TSP is

> "the problem of a salesman who wants to find, starting from his home town, a shortest possible trip through a given set of customer cities and to return to its home town, visiting each city exactly once."[1]

The problem is analogous to finding a minimal length Hamiltonian cycle in a graph, and can be formalized as a simple integer program. This problem is NP-complete (Garey and Johnson, 1979).

A setting that better applies to trucking companies is called the *Vehicle-Routing Problem*. It was first formulated by Dantzig and Ramser (1959) as

> "the problem of finding a set of routes with minimal overall cost for a fleet of vehicles based at one or several depots for a number of geographically dispersed cities or customers."[1]

Based on small modifications of the original definition, there is a plethora of different problems, such as the *Capacitated VRP*, the *Multiple Depots VRP*, the *Periodic VRP*, the *Split Delivery VRP*, the *Stochastic VRP*, the *VRP with Backhauls*, the *VRP with Pick-Up and Deliveries*, the *VRP with Satellite Facilities*, or the *VRP with Time Windows* (Toth and Vigo, 2002b). All these problems are NP-hard (Lenstra and Kan, 1981).

A common feature of all VRP problems is that every truck starts from a depot and returns to the same depot at the end of its path. Orders are represented by a single location (the customer)

[1] Definition is from the VRP Web site http://neo.lcc.uma.es/radi-aeb/WebVRP/

where the truck either drops off a package or collects one before returning to the depot. As a generalization of this property, the *General Pickup and Delivery Problem* (Savelsbergh and Sol, 1995) prescribes two stops for every order. The authors describe this problem as follows.

> "In the General Pickup and Delivery Problem (GPDP) a set of routes has to be constructed in order to satisfy transportation requests. A fleet of vehicles is available to operate the routes. Each vehicle has a given capacity, a start location and an end location. Each transportation request specifies the size of the load to be transported, the locations where it is to be picked up (the origins) and the locations where it is to be delivered (the destinations). Each load has to be transported by one vehicle from its set of origins to its set of destinations without any transshipment at other locations."

There are no depots associated to trucks, and every order is moved between its origin and destination, possibly via other locations. If there are time windows associated with the pickup and delivery locations, then the problem is called the *Pickup and Delivery Problem with Time Windows* (Mitrovic-Minic, 1998). Another variation is the *Dial-a-Ride Problem* (Feuerstein and Stougie, 2001), which is concerned with the transportation of humans (elderly, handicapped people, or taxi services). It adds special constraints to the original problem, like limited travel time for the travelers, or maximum travel distance.

The pickup and delivery problem with time windows is a rather general abstraction in the area of transportation problems. It covers a wide variety of real problems, but it does not include dynamic events that are of special interest in this thesis. For that we have to look into *dynamic transportation problems*. A pioneer of dynamic transportation problems, Psaraftis defines a problem to be *static* if all available information is revealed before the solution mechanism is required to produce results (Psaraftis, 1995). In contrast, if the solution mechanism must consider information that arrives after the production of the plans (and therefore is forced to reconsider these plans), Psaraftis calls the problem dynamic.

There are problem formulations that, in spite of containing uncertain information, are classified as static problems. For example, most of the stochastic vehicle-routing problems, such as the *Stochastic Traveling Salesman Problem* (Jaillet, 1985, 1991), are static problems according to this definition. Although they contain stochastic elements, they require the mechanism to produce a plan before the stochastic elements are realized. Therefore, the planning mechanism has no chance to react to the actual realizations of the stochastic values. The transportation problems, which we are interested in, belong to the group of dynamic problems (Gendreau and Potvin, 1998; Bianchi, 2000; Yang et al., 2004). Here, algorithms can react to changes, should the change add new information to the problem, or change the value of a parameter. We now review such dynamic transportation problems.

Just as the traveling salesman problem is at the core of all vehicle routing problems, the *Dynamic Traveling Salesman Problem* represents the base of dynamic problems (Psaraftis, 1988).

> "In the simplest version of this problem, demands for service are generated at each node of a network according to a Poisson process of parameter λ. The travel times between nodes are known and deterministic, and the salesman spends a known service time at each node. What is the routing policy that minimizes the average, over

> all demands, expected time until service of the demand is completed? Alternatively, what is the routing policy that maximizes the average expected number of demands serviced per unit time?"

That is, while the travel times and service times are given beforehand, new orders are generated during the travel of the salesman, who is required to minimize the time needed to serve all orders.

Many types of routing problems can be turned into dynamic problems in a similar manner. The common feature of these problems is that the demand is not necessarily known beforehand, but is revealed, step-by-step, while solving the problem. Dynamic problems of notoriety include *Dynamic Traveling Repairman Problem* (Bertsimas and van Ryzin, 1991), the *Dynamic Vehicle Routing Problem* (Psaraftis, 1995), the *Dynamic Pickup and Delivery Problem* (Swihart and Papastavrou, 1999), and the *Dynamic Dial-a-Ride Problem* (Feuerstein and Stougie, 2001).

Only a few researchers have studied sources of dynamism other than order release. Time dependent travel times appear to be the only other sources of uncertainty considered in the field of vehicle routing (Malandraki and Daskin, 1992). This source of uncertainty is studied by Ichouaa et al. and Woensel et al. as a stochastic (thus static) problem, in which the solution algorithm incorporates a stochastic model of dynamic travel times (Ichouaa et al., 2003; Woensel et al., 2008). Jung and Haghani consider it as a dynamic problem together with new order arrival (Haghani and Jung, 2005; Jung and Haghani, 2001). They model travel time and transportation demand changes throughout the day, allowing their algorithm to adapt the plans. In the related field of project scheduling, resource breakdowns are considered by Lambrechts et al. (2010), who suggest that inserting slack into the plan increases its robustness against resource breakdowns, and analyse different slack-insertion methods.

Finally, we mention the work of Figliozzi et al. on pricing in dynamic vehicle-routing problems (Figliozzi et al., 2007). They consider transportation companies bidding for orders sequentially, and analyze the problem of optimal bidding. Although this does not look like a transportation problem, the costs of the transportation companies are given by solutions to the underlying vehicle-routing problem. The authors formulate the problem as a *Vehicle Routing Problem in a Competitive Environment*, which shows how the company-oriented viewpoint of all these problems evolves to include the environment of the company.

2.1.2 Task-Allocation Problems

When human planners make plans and coordinate the execution of the plans, they are considering multiple interests. If a customer or a driver feels discrimination during execution, s/he will complain. In addition to considering people's feelings, planners still have to produce efficient plans (which is the interest of the management). Working in such an environment requires the planners to find a delicate balance between all these interests. Such situations can be modeled as multi-agent systems.

In fully distributed logistical multi-agent systems, every participant is modeled by an agent. Trucks and drivers are agents, customers, orders, and/or transportation companies are agents. Every agent has a goal it must reach. Truck and driver agents want to collect as many orders as they can; order agents want to find the cheapest transport; transportation company agents want to minimize the cost generated by all truck agents belonging to them, etc. In addition to agents

and their goals, agent systems specify the rules of interaction between the agents by providing communication protocols. Agents might bargain among each other, they may organize auctions, or use other kinds of coordination protocols. The important property of such systems is that the agents are autonomous, their information is hidden from other agents, and they share their private information via their interaction protocols. The goal of a logistical multi-agent system is to make contracts between truck and order agents, thereby *assigning* orders to trucks, in such a way that the individual goals of the agents are fulfilled.

Such an assignment problem can be seen as a *task allocation problem*.

> A task allocation problem is defined by a set of tasks T, a set of agents A, a cost function $c_i : 2^T \rightarrow \Re \cup \{\infty\}$ (which states the cost agent i incurs by handling a particular subset of tasks), and the initial allocation of tasks among agents $\langle T_1^{init}, \ldots, T_{|A|}^{init} \rangle$, where $\bigcup_{i \in A} T_i^{init} = T$, and $T_i^{init} \bigcap T_j^{init} = \emptyset$ for all $i \neq j$.

In a logistical problem, the tasks are the orders, the agents are the trucks, and the cost function is not additive. The cost of transporting two containers is not necessarily the sum of transporting them individually.

Task allocation problems are studied in relation to a wide range of problems from insects' social behavior (Hirsh and Gordon, 2001), through optimal network call routing (Dutta et al., 2006), to robotics (Lerman et al., 2006) (Gerkey and Mataric, 2004). Distributed software or hardware systems (Poladian et al., 2004) (Bataineh, 2005) (Sagar et al., 1989), sensor networks (Kuorilehto et al., 2005), scheduling (Sycara et al., 1991), or truck routing are also typical areas. An alternative formulation of this problem is called the resource allocation problem (Chevaleyre et al., 2005). In this case agents need resources, but resources have associated costs. The goal again is to allocate resources to agents with minimal possible costs incurred. For an in-depth analysis of the *Multi-agent Resource Allocation Problem* see (Chevaleyre et al., 2006).

Generally, task allocation problems can be categorized along multiple dimensions. First, a problem can be static or dynamic. Analogous to the static and dynamic transportation problems, a task allocation problem is static, if all the activities requiring resources are known before decisions have to be made. Consequently, if only a portion of the activities are known when allocation decisions have to be made, then the problem is called dynamic. A second dimension is the activity-resource relationship. An activity may require a single resource, or multiple ones, and resources may be assigned to a single activity, or to multiple ones. Problems can be categorized as one-to-one, one-to-many, many-to-one, or many-to-many along this dimension. Classification of task allocation solution approaches also follows the above-mentioned problem categories. Additionally, approaches have one more obvious, very important property: whether they are distributed or not. One approach is that of a central entity that has the authority to assign the tasks to resources and has the computational power to compute the allocation. Alternatively, distributed entities may try to coordinate to cooperatively calculate an efficient allocation (as is the case in a multi-agent formulation).

In comparison to operations research formulations, distributed task allocation models emphasize individual players, rather than the company perspective. Truck drivers, customers, and dispatchers are modeled as separate entities with their own goals. These goals can of course be aligned to optimize group (company) performance (maximize the joint utility), but it is not necessary. In such models, even if the players are supposed to optimize group utility, they must

find their way individually, using some coordination method (Jennings, 1996). This distribution of responsibilities is at the core of task allocation models, and it is what differentiates them from centralized models.

The main benefit of using multi-agent models for transportation problems is that they facilitate the modeling of the individual interests present in the problem. Defining agents to represent such interests (belonging to a single person or to a whole group) enables the model to consider multiple points of views, which is important in evaluating the 'logistical performance' of transportation companies. Additionally, the separation of concerns in the model enables a proper modeling of the information-sharing problem that the parties face. Certain agents may share information with others to achieve better performance, while other agents may keep their state secret; behavior witnessed in real-world companies. Finally, modelling conflicting interests a game-theoretical analysis is possible, which may help in understanding the effects of different behaviors.

In the following sections, we introduce some of the techniques used to tackle these problems. Of course the literature on algorithms is significant, therefore we can only attempt to list the most relevant ones.

2.2 Solution Approaches

This section reviews the different solution approaches that are associated with the different problem formulations of the previous section. First, we discuss exact and heuristic solution approaches to OR problems in Section 2.2.1. These approaches we call centralized approaches, because they assume a global view of the problem, and use all available information in finding a solution. In Section 2.2.2, we present the algorithms that are employed by multi-agent systems to solve transportation problems. We call these distributed approaches, because the decisions made in multi-agent systems are based on limited information, and are distributed by nature.

2.2.1 Centralized Approaches

Centralized approaches solve OR problems either exactly or approximately. The next section presents exact algorithms that aim to find an optimal solution to any given problem; the subsequent section discusses heuristics that search for local optimum solutions.

Exact Methods

The TSP and its derived vehicle-routing problems (the VRP, the PDP, and the DARP) are strongly interrelated problems. Approaches that are developed to solve one of these problems can be successfully applied to solve another one. Moreover, general combinatorial optimization approaches can be applied to any of these problems (for example direct tree search methods, dynamic programming (Bellman, 1957), integer linear programming methods (KulKarni and Bhave, 1985), or branch-and-bound techniques (Toth and Vigo, 2002a)). It is not our goal to introduce all of the exact solution approaches that were proposed for these problems. Instead we concentrate on approaches for the PDP. For an overview over the TSP and VRP, we refer the reader to Lawler et al. (1985) and Laporte (1992) respectively.

Early work on dial-a-ride problems, as a special case of pickup-and-delivery problems, targeted single-vehicle immediate-request dial-a-ride problems and single vehicle dial-a-ride problems with time windows (Psaraftis, 1983, 1980). In this early work a straightforward dynamic program was used to calculate the optimal solution. Later, Desaulniers et al. introduced a branch-and-bound approach based on set partitioning combined with the Dantzig-Wolfe decomposition technique, a.k.a. column generation (Desrosiers et al., 1984). This decomposition technique produces very tight bounds for branching, therefore the branch-and-bound tree is kept reasonably small. Later, they successfully applied it to pickup and delivery problems (Dumas et al., 1991). A typical example of reusing results achieved for the TSP is the following approach by Desrosiers et al., who addressed a multiple-depot, full-truck-load pickup and delivery problem by transforming it to an asymmetric TSP (Desrosiers et al., 1988).

The dynamic variants of these pickup and delivery solutions (and in fact, most other vehicle-routing solutions), are based on the iterative evaluation of the static versions. Psaraftis, for example, extends his static dynamic-programming algorithm for the immediate-request single-vehicle dial-a-ride problem in this manner (Psaraftis, 1980). The dynamic program is invoked repeatedly, whenever a new order arrives. For the dynamic multiple-vehicle pickup and delivery problem, Psaraftis develops a rolling-horizon approach (Psaraftis, 1988). Only those orders, for which the earliest pickup time is closer than a predefined L (the length of the horizon), are considered by the vehicles. Even then, orders are only tentatively assigned to vehicles, until their earliest pickup time is closer than αL for some $\alpha \in (0,1)$. Powell et al. study the dynamic full-truck-load pickup and delivery problem (among others), but they use a stochastic network model (Powell et al., 1995). They consider future requests by stochastic network links, and as such, they solve a static routing problem. Yang, Jaillet, and Mahmassani describe a rolling horizon approach for the dynamic truckload pickup and delivery problem with time windows (Yang et al., 1999). Requests arise continuously while trucks are serving customers. Previously assigned (but not yet executed) requests can be reassigned, and trucks can be deviated on their way to the current pickup location at every decision point. When new requests arrive, the assignment is re-evaluated via use of a mixed integer program. They compared their solution approach to simple rule-based approaches, and found that up to a certain point, it pays off to consider all available information.

Since the underlying problems in vehicle routing are generally NP-hard, exact methods may need a considerable amount of time to produce results. Therefore, a wide variety of heuristic approaches has evolved to supply timely solutions. The next section provides an overview of these methods.

Heuristics

There are two main categories of heuristic algorithms for vehicle routing problems. Early research concentrated on capitalizing on the structure of routing problems in deriving *classical heuristics*, while more recently researchers developed more general search methods called *meta-heuristics*.

Classical heuristics can further be divided into *clustering techniques*, *constructive methods* and *improvement methods*. Clustering techniques are composed of two phases (they are also called 2-phase algorithms). The clustering phase assigns customers to vehicles, and the routing phase generates routes for the vehicles. There are two flavors of such algorithms derived

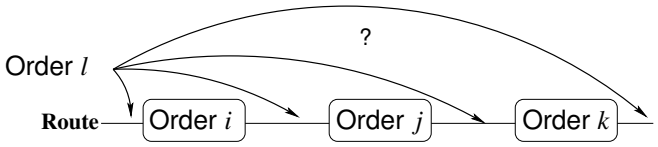


Figure 2.1: Insertion of a new order into a route.

from the possible orderings of the two phases. Route-first, cluster second algorithms generate a giant tour including all customers first, and then partition this tour according to the given constraints. Cluster-first, route-second algorithms reverse this order. Well known examples of this second order are the Fisher and Jaikumar algorithm (Fisher and Jaikumar, 1981), the sweep algorithm (Gillet and Miller, 1974), or the petal algorithm (Ryan et al., 1993).

Constructive methods build a feasible solution considering the given constraints and the objective function, but they do not try to improve it. One of the best known constructive method is the Clarke and Wright savings heuristic (Clarke and Wright, 1964). It starts with n back-and-forth routes between the depot and a customer. Then, in every iteration, two routes are merged by concatenating them and leaving out the depots from the middle of the new route. Such a merge is performed if the new route saves costs compared to the original one. In the parallel version, the merge yielding the largest saving is implemented, while in the sequential version, one route is extended as long as possible. A possible extension of this method is the use of a matching algorithm to optimize the merging process (Desrochers and Verhoog, 1989; Altinkemer and Gavish, 1991).

The *insertion heuristic* is the most relevant constructive heuristic for this thesis, because it is widely used in distributed agent-based models. It builds routes by inserting customers into the routes one-by-one. The defining property of this method is that sequences of already assigned orders are never reordered. New orders are always inserted either at the beginning of a route, at the end of it, or between two already inserted orders (See Figure 2.1). Insertions can be sequential or parallel. In sequential insertions, new orders are inserted into the same route as long as it is feasible. A new route is only considered when the previous route is full. In the parallel version, insertions into all routes are considered for every new order, and the best insertion is chosen. An important property of the method is its sensitivity to the choice of the next order to be inserted. Possible choices include taking the order that is cheapest to insert, or the one that is farthest from the depot. In some applications, the order of insertion might be given *a priori*. In general, the algorithm requires $O(n^3)$ steps, but if additional constraints need to be considered, this changes to $O(n^4)$ (Campbell and Savelsbergh, 2004).

The constructive methods discussed above are usually used to provide an initial solution for improvement techniques. Improvement techniques generally start from an initial solution, and try to improve on it. Thompson and Psaraftis introduce cyclic transfers of orders among vehicles (Thompson and Psaraftis, 1993). A *b-cyclic k-transfer* shifts k orders from each vehicle to the next one in a circular permutation of b vehicles. This is a very general framework that can express many different and complicated exchanges of orders between vehicles. As a result, the space of cyclic transfers is too big to consider a full search. Followers (Breedam, 1994; Kinderwater and Savelsbergh, 1997) have tried to introduce simpler moves that can be summarized as follows:

Customer Relocation moves a single customer or multiple customers from one vehicle to another.

Customer Exchange exchanges a single customer or a sequence of customers between two vehicles.

Crossover mix the routes of two vehicles in such a way, that after certain customers, the vehicles continue servicing the customers of the other vehicle.

Once a route has been constructed (the initial solution), there is a set of neighboring solutions that can be reached by the application of one such moves. Classical heuristics search the full neighborhood and choose the move that leads to the best neighboring solution. This way, through repeated application, an algorithm searches for the optimal solution by jumping from one locally optimal solution to another.

Meta-heuristics apply classical heuristics in various ways. A common feature of these techniques is that they allow moves to worse solutions (sometimes even to an infeasible one) in certain circumstances. *Simulated annealing*, for example, moves to a worse solution with a certain probability. This probability (the temperature) decreases as the search moves on, thereby simulating an annealing process. A similar technique is used in *deterministic annealing*, except that there, worse solutions are always accepted if they are below a threshold (Dueck and Scheurer, 1990; Dueck, 1993).

When allowing moves towards a worse solution, mechanisms must be implemented to avoid going in cycles. After moving to a worse solution, improvement steps might lead back to the same local optimum that you tried to escape. *Tabu search* algorithms maintain a list of the properties of already visited solutions to exclude new solutions that possess the same properties (Glover, 1986). Since the search considers only a subset of the neighborhood in every step, it can move to a worse solution, but the application of the tabu list prevents it to return to the same track. The neighborhood moves used in tabu search are usually those introduced in the classical improvement methods, but methods have been enhanced by specialized diversification and intensification strategies for guiding the search (e.g. Toth and Vigo, 2003; Cordeau et al., 2001b; Taillard et al., 1997; Kelly and Xu, 1996; Rochat and Taillard, 1995).

Ant algorithms, another example of a meta-heuristic, simulate movements in an ant colony composed of the locations that vehicles have to visit (Bullnheimer et al., 1997; Tian et al., 2003). Every ant leaves a pheromone trail while trying to construct a feasible solution. The strength of the trail is determined by the quality of the solution the ant produced. The algorithm runs for a fixed number of iterations. In every iteration, the ants first produce feasible solutions, then an improvement method optimizes the solutions, and finally the heuristic updates the trails.

Another meta-heuristic applied to VRP is *constraint programming* (CP) (Shaw, 1998). In CP, decision variables encode the routes of vehicles, and time, capacity, etc. constraints can be expressed as linear inequalities. The search for a solution is usually performed by a complete search technique such as depth-first search, while for the best solution usually a branch-and-bound method is applied. CP enhances the search using constraint propagation. If bounds or constraints on a variable can be inferred, or are tentatively set, these changes are “propagated” through all the constraints to reduce the domains of constrained variables.

Finally, the most widely used meta-heuristic algorithms in vehicle routing are probably *genetic algorithms* (GA) (Plum and Ali, 2005). Genetic algorithms work with a population of candidate solutions instead of just a single solution, in this way these algorithms make a multiple-way search simultaneously. Each individual represents a potential solution for the problem. The creation of a new generation of individuals involves three major steps or phases. (1) The selection phase consist of randomly choosing two parent individuals from the population for mating purposes. The probability of selecting a population member is generally proportional to its fitness in order to emphasize genetic quality while maintaining genetic diversity. Here, fitness refers to a measure of profit, utility or goodness to be maximized while exploring the solution space. (2) The recombination or reproduction process uses the genes of selected parents to produce offspring that will form the next generation. (3) The mutation consists of randomly modifying the gene(s) of a single individual to further explore the solution space and ensure, or preserve, genetic diversity. The occurrence of a mutation generally appears with a low probability.

One feature of all the heuristic approaches mentioned above is that they can be naturally applied to dynamic problems. Let us consider the simple insertion heuristic. In a dynamic problem, where orders arrive one by one, it is natural to insert them into routes one by one, as they arrive. Since most of the improvement methods and meta-heuristics obtain their initial solution via the insertion heuristic, these algorithms can easily accommodate new orders as follows. Whenever a new order arrives, the improvement steps are stopped, and insertion is applied to the new order. Then the improvement steps continue. Such an approach to dynamic vehicle-routing problems is possible, because the improvement- and meta-heuristics are relatively insensitive to changes in the problem.

The most popular meta-heuristics applied in this manner are genetic algorithms (Djadane et al., 2006; Housroum et al., 2006; Jih and Yung-Jen Hsu, 1999). They solve dynamic problems similar to static problems, but in between the iterations, they can modify fitness functions and values to reflect the current situation. Another application of genetic algorithms in vehicle routing is provided by Alvarenga et al. (2005). They use it in their column generation technique to generate possible routes for the branch-and-bound algorithm (similarly to (Chen and Xu, 2006)). They claim that the genetic algorithm can easily incorporate the addition or cancellation of requests, while the set partitioning problem solver can still use the routes generated before the change in hopes of a reduced overall cost.

A special kind of heuristic approach is a *dispatching heuristic*. These simple rule-based methods are often used as a bottom line comparison for algorithms for dynamic vehicle-routing problems (Powell et al., 1995). They usually assume that there is not enough information to produce durable plans, therefore they define simple rules to vehicles regarding what they should do after servicing a customer. Examples of such rules can be ‘drive to the closest customer’, or ‘drive to a local center and wait’, or alternatively ‘wait at the customer’.

Whether meta-heuristics, classical heuristics, or even exact approaches, all of the solution methods for static or dynamic transportation problems mentioned so far assume that all information regarding the problem can be gathered and made available to a central decision maker. These approaches take a global point of view and consider only the transportation company’s perspective when searching for a solution.

In reality, the transportation of goods is executed by multiple parties. There are customers

who submit orders, drivers who deliver goods and an environment that accommodates the whole operation. One can look at a transportation problem as a game played by multiple players all with their own interest. Approaches that solve these distributed models are discussed in the next section. We show how transportation problems modeled by distributed agent systems can be solved, and what techniques and solution methods have been proposed in the literature.

2.2.2 Logistic Multi-Agent Systems

Multi-agent systems (MAS) consist of distributed decision making entities (agents) that are capable of communication and cooperation, but follow their own goals (Wooldridge, 2002). The agents may share a common goal, or they may have different goals that they try to reach selfishly. Such models seem to be especially well suited for the transportation domain (Fischer et al., 1995). Fischer et al. have identified the following reasons to support this claim.

- The transportation domain is inherently distributed, therefore it is natural to look at it as a multi-agent system.
- Planning of transportation requests is a dynamic process, and multi-agent planning is especially aimed at such problems.
- Transportation companies are reluctant to share operational information that is required for a centralized planner. Multi-agent systems can very carefully model such situations, and share only necessary information.
- Modeling of the existing cooperative processes by task decomposition and task allocation are natural in multi-agent systems.

In addition to the special properties of the transportation domain that make the application of MAS suitable, the following points deserve emphasis (by partly repeating what has already been said).

- A multi-agent system distributes the decision making process, which might be crucial in handling frequently occurring changes. It allows simultaneous handling of the incidents which can contribute to the timely reactions, and therefore to robustness.
- Agents in a MAS have their own goals and responsibilities, which makes it simple to include local constraints in the agents to truly represent real-life players.
- Agents represent entities with different interests, and their interaction can be analyzed withing the framework of game theory. Incentives of the players can be considered, and new rules may be designed to motivate the players to behave in a more efficient way.

In comparison to centralized OR approaches, multi-agent systems apply the same algorithms in a distributed manner. They shift the focus from the centralized company perspective to a distributed network-based perspective of the individual players. This can be regarded as an extension of the centralized models that attempt to include the environment of the companies into the model.

In the following we discuss multi-agent methods for various transportation problems. They all use some kind of task-allocation technique, and thereby distribute decision making. They differ, however, substantially in the problems they address, in exactly what is modeled by each agent, and in the negotiation protocol used for task allocation. They are all relevant to this thesis, because they all address dynamic transportation problems.

Classical Agent Heuristics

In 1995, Fischer et al. described a multi-agent solution for the dynamic vehicle-routing problem with time windows (Fischer et al., 1995). They defined two types of agents. Company agents to coordinate the allocation of orders to trucks, and arrange swaps with other companies for ill-fitting jobs. On a lower level, truck agents consider only the routes of single trucks. They belong to company agents, who use their Extended Contract Net Protocol (ECNP) to assign requests to trucks. The Contract Net Protocol was first introduced by Smith as a coordination protocol in a distributed problem solver (Smith, 1980) (Smith, 1981). It can be regarded as a sealed-bid first-price auction. In the original protocol, the user who wants to execute a task sends a request-for-quote to all task executing agents. After a certain amount of time elapses, the user evaluates all of the answers received, and chooses the best bid. The winning bid is confirmed and the others are rejected. This simple protocol was extended by Fischer et al. to accommodate partial bids. Here, agents can send bids for a part of the offered task. The auctioneer chooses the best partial bid and replies with temporal confirmation and temporal rejection messages. Then the auctioneer tries to contract out the remaining part of the task in successive rounds. If he manages to sell the whole task, he sends a definitive confirmation, otherwise he sends reject messages to the agents. In response to the request-for-quote messages, trucks use a polynomial time insertion algorithm to compute the cost of transporting the offered request and bid accordingly in the ECNP. After an initial allocation has been made, a simulated trading method is used to enhance the solution and also to incorporate changes in orders (Bachem et al., 1996). The model appropriately differentiates between inter-company bargaining and intra-company task allocation, and distributes local decisions to as low a level as possible. Although they have developed a model that can handle dynamic input and changing travel times, the authors did not test the model on dynamic problems. In their experiments they used static VRPTW instances and reported only on the optimality of the model. Unfortunately, they did not analyze the behavior of the model in different dynamic situations.

Holonic Agents

A similar approach was used by Bürckert et al., who followed a *holonic agent* approach to model a dynamic pickup and delivery problem (Bürckert et al., 2000). A holon is a group of agents that cooperate closely. Their agents represent physical entities such as trucks, chassis, containers, drivers, etc. The agents have to form holons in order to be able to accept and execute orders. Order assignment is coordinated by company agents topping the hierarchy. The procedure is similar to that of Fischer et al. An extended contract net protocol is used to assign the requests to holons. Every holon makes its own plan and submits bids. After the initial assignment a simulated trading procedure is used to enhance the solution. The authors described a distributed agent approach for the dynamic PDP, but they did not provide any experimental results to assess

its performance. Their main focus was on enabling humans to produce plans together with agents, and the generated plans were not evaluated in comparison to other approaches. Consequently, the dynamic behavior of their model is unknown.

Parallel Bidding Methods

Another followup on Fischer et al.'s article is the work of Perugini, Lambert, Sterling, and Pearce (2003a). They extended the extended contract net protocol to solve the eager bidder problem (Schillo et al., 2002). In their approach, they allow agents to bid on multiple tasks without being committed to the task. Thus agents can reply to parallel request-for-quotes in an effort not to miss any good deals. To enable this they have introduced additional withdrawal messages by which any party can cancel a bid. The authors achieved better results in terms of optimality, compared to the previous ECNP, but the protocol, again, is not tested against a changing environment.

A recent extension is introduced by Aknine, Pinson, and Shakun summarizing previous ECNP extensions, and adding failure detection abilities to avoid deadlock situations caused by malfunctioning agents (Aknine et al., 2004). This solution also handles so called *decommitment* situations (withdrawal in the previous model), when one party wants to break an already existing contract.

Information Centralizing Methods

A model focusing on real-world applicability was introduced by Dorer and Calisti (2005). They implemented an approximation method for the dynamic pickup and delivery problem with soft time windows using a multi-agent system. The approximation first uses an insertion method to set up feasible routes and then improves the routes via cyclic transfers. Technically, in a b-cyclic k-transfer k orders are cyclically exchanged among b-routes. Although they used agents, they did not model trucks and orders as agents. Instead, their agents stand for (geographically disjunctive) clusters of trucks. This model represents a compromise between a totally centralized and a totally distributed system. The authors emphasize their system's ability to handle real-world problems of 3500 orders, but they fail to analyze its flexibility. Their order set is approved by human planners as being realistic, but it is unknown, how difficult this set is to solve or how dynamic those orders are.

There are also other attempts that try to compromise between fully distributed and fully centralized systems. Leong and Liu use a fully centralized optimizer to initialize the agents (Leong and Liu, 2006). The agents' role is then to change the plans as events are revealed. The authors analyze the performance of their model on a selection of problems from the Solomon benchmark sets, and show that it performs competitively. The same approach is analysed by Davidsson et al. (2007b) on a production-transportation-inventory problem with dynamic demand. The authors compare the solutions of different combinations of agents with and without embedded optimization, and global optimization. They conclude that global optimization is competitive when the input is predictable, while agents with embedded optimization initialized by a globally optimal plan perform best when input is unpredictable. In a more distributed approach, Mes et al. (2007) use a hierarchy of agents, but with a different role than Fischer et al. (1995). Their two high-level agents (the planner and the customer agent) gather information from and provide information to agents assigned beneath them (truck and order agents). The

role of the higher level agents is to centralize information essential for the lower level agents to make the right decisions. At the decentralized end of the spectrum, Kohout and Erol introduce an improvement process that works between agents (Kohout and Erol, 1999). The process mimics an improvement technique called 'swapping' or 'two-exchange' (Cordeau et al., 2001a). They implement this swapping process in a fully distributed way, and show that it yields significant improvement. In a very different approach, Holmgren et al. (2009) use a multi-agent approach to distribute a Dantzig-Wolfe decomposition of a transportation problem (a.k.a. column generation). They argue that the subproblems can be delegated to planning agents, while a coordinator agent can take care of the master problem. The advantages of such a solution method for a mixed-integer problem are the speedup obtained by simultaneously solving the subproblems, and the increased confidentiality obtained by keeping confidential information at the planning agents as opposed to centralizing all information at the coordinator agent.

Decommitment in Vehicle Routing

A different approach is taken by Hoen and Poutré, who applied the idea of leveled commitment contracts (Sandholm, 1999) to a transportation problem (Hoen and Poutré, 2003). When leveled commitment contracts are in effect, agents are allowed to break contracts, if they are willing to pay a predefined penalty. This allows agents to exchange previously accepted contracts for new more-beneficial contracts if they are better even after paying the penalty. Agent behaviors regarding decommitment can follow a myopic strategy, decommitting whenever they need to. Alternatively, agents can be strategic by not decommitting and waiting for the other party to decommit. If an agent expects the other party to decommit, strategic behavior may have a higher expected payoff, due to the collection of the decommitment penalty. Andersson and Sandholm compared the two different behaviors empirically, and also analyzed different methods for setting the penalty (Andersson and Sandholm, 1998). They have concluded that it is best to set the penalty low, but higher than zero, and that in such settings, myopic agents perform almost as good as strategic agents.

Sandholm and Lesser gave a formal proof that even if strategic decommitment takes place, leveled commitment contracts enable more profitable deals than full commitment contracts (Sandholm and Lesser, 2001). Sandholm, Sikka, and Norden took the theory further by analyzing decommitment games (Sandholm et al., 1999). They provided algorithms to find the Nash-equilibrium decommitment threshold and decommitment probabilities for a given contract, and to find the contract that maximizes the expected payoff even in case of insincere decommitment. These algorithms use game-theoretical analysis to find the optimal solution for rational agents.

A more practical approach was taken by Brandt and Weiss. They analyzed the combination of auction protocols and leveled commitment contracts. They experimented with English auctions (Brandt and Weiss, 1999), with Dutch auctions (Brandt and Weiss, 2000), and with Vickrey auctions (Brandt et al., 2000). Based on their initial efforts, they concluded that combining auction protocols with leveled commitment contracts results in a flexible assignment scheme that shows desirable fairness properties with respect to the profit made by the contractees. Their results are promising, because such fair mechanisms are very attractive for multi-agent systems, where agents are self-interested, such as the humans they represent. Fairness can be a key element for making participation in such a system rational for many human players.

Hoen and Poutré modeled a single-company case, where trucks could re-allocate orders via decommitment. In their approach, an order is initially assigned to a truck by the Vickrey auction protocol (Vickrey, 1961). It is a one-shot closed-bid auction, where players send their one and only bid in a closed envelop to the auctioneer. The auctioneer selects the highest bid as the winner, and the price is defined as the highest submitted bid without the winning bid. This auction type is particularly interesting for AI researchers, because the dominant bidding strategy of the players is to bid their true value. This property is called *incentive compatibility*. Although humans may not particularly like this property, in computer systems it is very popular, because it undercuts attempts of counter-speculation from other agents. Simply put, computer agents do not have to engage in lengthy reasoning about the opinion of the other agent, recognising that the other agents also opinion about the primary agent. Knowing their own valuation, computer agents can simply derive their bids without further calculations. Accordingly, the Vickrey auction protocol is part of many research efforts addressing agent coordination (Brandt et al., 2000; Larson and Sandholm, 2004; Likhodedov and Sandholm, 2003; Rosenschein and Zlotkin, 1994). In addition to its popularity, the application of the Vickrey protocol has also received criticism (Sandholm, 1995). These include the need to trust the auctioneer about the value of the second-best bid, and side constraints that need to be satisfied in order for the incentive compatibility property to hold.

Hoen and Poutré applied Vickrey auctions and decommitment to a truckload vehicle-routing problem, and showed that decommitment yields more efficient plans for the trucks, but they did not analyze different dynamic settings.

By this overview of logistical multi-agent models, it is apparent that numerous agent models have been developed for dynamic transportation problems. Even though these models usually address dynamic transportation problems, evaluation of the models on dynamic instances is rare. The models are either compared to other methods based on static problem instances, or not compared at all. A similar conclusion has been drawn by Davidsson et al. (2007a), who analysed 28, not exclusively transportation related papers published in the proceedings of the MABS workshop series, and found that agent-based models are rarely compared to previously developed models, and that implementation details on how the results were obtained are generally missing. This is an area the current thesis seeks to contribute to by providing an evaluation framework in details to compare dynamic transportation problems.

2.2.3 Robust Vehicle Routing

In the vehicle routing literature, we can find two distinct interpretations of robustness. Sometimes robustness means that a routing method can handle a wide range of routing problems equally good, or at other times, it can refer to the performance of the routing method when faced with stochasticity and/or dynamism.

The former interpretation is used by Fukasawa et al. when they combine two solution methods: column generation and cutting panes, to solve previously unsolved, capacitated vehicle-routing problems (Fukasawa et al., 2006). Each of these methods were successfully applied before to distinct subsets of vehicle routing problems, and the authors argue that the combination of them provides a robust method that can solve problems previously not solved.

The other interpretation of robustness relates to the performance of routing methods when faced with stochastic and/or dynamic elements in the problem. A basic example of such robust

algorithms is provided by Bell et al., who developed a time-dependent algorithm for robust vehicle navigation in time-dependent stochastic road networks (Bell et al., 2010). In their stochastic setting, they search for a shortest path on a time-dependent network, with the goal of minimizing expected arrival times.

Using the same meaning of the word robustness, there are a couple of proposed solutions for dynamic and/or stochastic vehicle routing problems that are based on the work of Ben-Tal and Nemirovski on robust optimization (Ben-Tal and Nemirovski, 1998). Sungur solves a VRP with demand and travel-time uncertainty using the method of Ben-Tal and Nemirovski, and concludes that the distribution of slack in the vehicles over the network is the key to success. Additionally, they conclude that a problem with clustered demands far from the depot with a dense random zone near the depot facilitates the collaboration of the vehicles to distribute their slack, and thereby yields more robust solutions.

Erera et al. solve a vehicle routing problem with stochastic demand and duration constraints using a similar, robust-optimization method. In this problem customers are known, but the amount of demand at each customer is unknown. Additionally, the problem is burdened with hard route-duration constraints for the vehicles. The paper proposes a solution method based on the combination of robust optimization and three different recourse actions. Their computational experiments shows that vehicle routes and expected costs can vary significantly due to the hard route-duration constraints.

In addition to the robust optimization-based solutions, Bertsimas and Simchi-Levi survey robust vehicle routing algorithms that address uncertainty in demand, travel time, and service times at customers (Bertsimas and Simchi-Levi, 1996). They present a comprehensive set of models for such problems, and discuss different methods that dispatch vehicles to jobs while minimizing the number of pending jobs in the system.

The above mentioned methods are usually evaluated in a worst-case scenario. In case of the robust-optimization methods, for example, a robust optimal value is found by assuming a pessimistic setting for the parameters that are uncertain. For such methods, the difference of the strict optimal and the robust optimal values, called as the cost of robustness, is also very informative. In an analytical discussion of solution methods, such as the one by Bertsimas and Simchi-Levi, it is similarly usual to define the worst-case performance of the method, possibly along with the average-case performance. When analysing a heuristic or approximation algorithm, however, we can also define its precision using a competitive-ratio measure (Klein, 1992), which is the ratio of the solution found by the heuristic and the optimal value. A similar measure, called the amortized efficiency, is used by Sleator and Tarjan (1985) in an experimental setting, in which the authors average the competitive ratios of different algorithms over a set of worst case scenarios. Such a measure, obtained through experiments, show how the performance of an algorithm degrades for different inputs.

In the previous sections, we reviewed different formulations of vehicle-routing and task-allocation problems, as well as centralized optimization- and heuristic-based, and distributed agent-based solution methods for these problems. In this section, we gave a quick overview of research that considers the robustness of algorithms. In the next section, we look at how transportation is judged by professionals, and how this compares to the academic models.

2.3 Performance Indicators

Transportation is a highly competitive industry with a lot of participants. No two companies are exactly the same, and they all define their own business processes and evaluations. Researchers in the field of *Supply Chain Management* have been developing frameworks for evaluating complex processes. An overview of supply chain performance measurement has recently been published by Shepherd and Günter (2006). Earlier, Beamon criticized performance evaluation models for using oversimplified measures (Beamon, 1998, 1999). He identified three main areas that should be considered by supply chain designers: efficiency, customer service level, and flexibility to respond effectively to a changing environment.

A similar conclusion was drawn by Krauth, Moonen, Popova, and Schut after interviewing several logistics service providers about their performance measurement techniques (Krauth et al., 2005). They gathered more than one hundred different performance indicators used by these companies to assess their 'logistical performance'. The authors categorized the indicators, or the properties they measured as short or long term indicators, and along another dimension they grouped them into four sets. The four sets are labeled according to the interest group, to which the indicators belong. The *'management'* set contains cost related indicators, such as travel distance, empty distance, and short term costs of lateness, job rejection, etc. The *'employee'* set contains indicators that employees (truck drivers in their case) care about. Such variables are, for example, overtime work, frequency of breaks (idle times), if they have to go to locations they do not like, etc. The third set encompasses *customer* satisfaction indicators. Late deliveries, non-appropriate handling, and value-added services belong to this set. Finally, they identify a set that contains performance metrics that are important to a broader circle, i.e. to *society*. Their conclusion is that performance measurement of logistical activity should contain elements of all four groups, as well as long and short term indicators.

Methods taking the global point of view (discussed in Section 2.2.1) are usually concerned with performance indicators in the 'management' set. Distance-based cost metrics can be easily incorporated in the objective function of optimization algorithms. In some cases indicators from the 'customer' group are translated into operational costs (lateness or rejection penalties) and incorporated as well. Even then, only short term effects of being late or rejection is considered, ignoring the long term effects of such behavior. Other indicators, however, can hardly be seen in optimization models. Sometimes some of the aspects are considered as additional constraints, but usually it is difficult to formalize them appropriately (Xu et al., 2003). While studies show that real-world companies tend to take a broader point of view in evaluating their performance, the previously mentioned approaches support only a subset of the evaluation criteria.

In addition to considering only a subset of all involved parties, the global methods assume that all information can be gathered to make the best decision. This may cause privacy issues in real life. It might be difficult to retrieve all important information simply because the owner of the information would not like to publish it. Methods that consider a local point of view (see Section 2.2.2) seem to be better suited to such situations, since they do not assume having complete information.

2.4 Summary

In this chapter, we reviewed the literature of two approaches to dynamic transportation problems. These two are the centralized optimization and the distributed agent approaches. First we presented an overview of existing problem definitions related to transportation problems in Section 2.1. We discussed the most important vehicle-routing problem formulations, and especially those describing dynamic problems. In addition to these classical OR problem definitions in Section 2.1.1, we discussed task allocation problems in Section 2.1.2, which capture similar problems and are more apt for a distributed solution method. In all dynamic problem formulation described here, dynamism is given by new order arrivals during the execution of plans. This reflects the general state of the literature, where this kind of uncertainty is by far the most studied. Other dynamic events such as truck breakdowns, service- or travel time uncertainties are minimally studied in the literature.

Subsequent to introducing the problem formulations, we introduced the most important solution approaches in Section 2.2. We described exact and heuristic solution approaches to vehicle-routing problems in Section 2.2.1. Heuristics are important not only because they can solve such problem formulations, but also because they provide the foundations of distributed methods. Then in Section 2.2.2 we showed how distributed multi-agent approaches solve routing problems formalized as task-allocation problems. We identified 'decommitment' as a very promising technique to handle dynamic changes. Unfortunately, the application of this technique in vehicle-routing problems is very limited, especially the optimal level of 'decommitment penalty' in dynamic settings is an open question.

One issue identified in the multi-agent literature of transportation problems is the inappropriate comparison of new methods to existing ones. In several cases solutions developed for dynamic cases are compared on static instances, due to missing benchmarks of dynamic problems. Indeed, measuring logistical performance in dynamic situations is complex. Section 2.3 discusses briefly how real companies evaluate their performance.

To summarize the conclusions of the literature review, we identify three main points that will receive further attention in this thesis. First, in current formulations of dynamic transportation problems, the only source of dynamism is the arrivals of new orders. Other sources such as service- or travel time uncertainty are especially interesting because they represent a different kind of uncertainty: one that changes certain parameters of the problem, rather than introducing new elements in the problem. Second, the technique of 'decommitment' is a promising tool that can be used in multi-agent approaches, but the proper setting of the penalty level in dynamic problems is an open question. Third, the technology of comparing approaches on dynamic instances needs improvement.

The next chapter makes the first steps in providing a comparison or evaluation framework by defining a transportation problem with uncertainties, in conjunction with a robustness measure that will be used to compare different methods.

Chapter 3

Problem Description and Evaluation Criteria

3

To compare different logistical planning approaches, one needs three things: a transportation problem, a set of solution methods to compare, and a method to compare the results of these approaches. The goal of this chapter is to define the transportation problem and provide two different models for the defined problem. Additionally, different measures quantifying the performance of solution methods are analysed here. The methods that solve these models along with experimental results are presented in Chapters 4-6.

Our main interest in this thesis lies in the dynamic behavior of logistical planning methods; we study how different planning methods behave under uncertainty. To this end, we first formalize a static container-transportation problem based on a real-world case in Section 3.1. Then in Section 3.2, we show how this problem can be augmented with different kinds of uncertainties. Several problem definitions are given with different kinds of uncertainties, and a problem with mixed uncertainties is also defined.

In Section 3.3, in addition to defining the problem, we provide two very different formulations for this problem: a distributed and a centralized one. A comparison of these formulations (and their associated solution methods) was published in Máhr et al. (2010). That comparison was based on a classical cost measure, but in Máhr et al. (2011), the study was extended to include a new robustness measure. This measure is discussed in Section 3.4

3.1 The Static Problem

In order to compare logistical planning methods on dynamic problems, we need to define a problem with dynamic events. First, in this section, we formally define a Pickup-and-Delivery with Time Windows problem, which is based on the operations of a transportation company that transports containers arriving at the port of Rotterdam.

Definition 3.1 describes a problem where a set of vehicles (K) need to deliver a set of orders (N), such that one vehicle can transport only one order at a time. In the real-world container transportation problem, each order consists of three locations, and for each location there is a waiting time and a time window given, such that the vehicle transporting the given order has to visit the three locations in order, it has to arrive at the locations within the assigned time windows, and at every location it has to wait for the specified amount time. The problem is solved by assigning the $|K|$ vehicles to $|K|$ tours, each of which is represented by a list of location-time-point pairs where the first and the last locations are the home location of the truck. The objective is to find tours such that the sum of time spent traveling empty (i.e. traveling from

home to the first order, traveling between the last and the first locations of orders, and traveling to home from the last order), the rejection of orders, and the lateness (the amount of time by which any of the time windows are violated) is minimized. Order rejection is penalized by the *loaded distance* of the rejected order, which is the total time needed to serve the given order (i.e. the waiting times plus the travel times between the three locations). This loaded distance represents the revenue lost by not transporting the order.

Definition 3.1

Name: *The Truckload Pickup-Delivery-and-Return Problem with Time Windows (1-PDRPTW)*

Instance:

- *a set L of locations with distances in the time domain defined as* $d : L \times L \to \mathbb{Z}$. $d(l_1, l_2)$ *denotes the time needed to travel between locations* l_1 *and* l_2*;*
- *a set K of vehicles, where each* $k \in K$ *has a starting location* $l^k \in L$*, and can transport a maximum of one order at a time;*
- *and a set N of orders, where each* $n \in N$ *consists of three stops* (l_1^n, l_2^n, l_3^n), $l_i^n \in L$ *that must be visited by the same vehicle in the given order, and for each stop a waiting time* w_i^n, $i \in \{1,2,3\}$ *denoting the time length the vehicle transporting the order must wait at* l_i^n*, and a time window* $[s_i^n, e_i^n]$, $i \in \{1,2,3\}$*, within which the truck must arrive at* l_i^n.

Solution: *a set P of tours (paths), one for each* $k \in K$*, and a set* $Y \subseteq N$ *of rejected orders. A tour* $p^k \in P$ *is a list of location-time* (l_i^k, t_i^k) *pairs, where* $i = 0..3m^k + 1$, $m^k \leq |N|$, m^k *being the number of orders assigend to vehicle k, such that* $\forall i \bmod 3 = 1 : \exists n \in N \wedge n \notin Y$*, and*

- $l_0^k = l^k$, $t_0^k = 0$,
- $l_i^k = l_1^n$, $t_i^k \in [s_1^n, e_1^n]$,
- $l_{i+1}^k = l_2^n$, $t_{i+1}^k \in [s_2^n, e_2^n]$,
- $l_{i+2}^k = l_3^n$, $t_{i+2}^k \in [s_3^n, e_3^n]$,
- $l_{3m^k+1}^k = l^k$,

and that $\forall i = 0..3m^k$

- $t_{i+1}^k - t_i^k \geq d(l_{i+1}^k, l_i^k)$.

Measure: *empty-travel time, rejection penalty, plus lateness, i.e.*

$$\min \sum_{k=1}^{|K|} t_e(k) + \sum_{y \in Y \subseteq N} p(y) + \sum_{k=1}^{|K|} t_l(k)$$

where

- $t_e(k) = \sum_{i=0}^{m^k} d(l_{3i}^k, l_{3i+1}^k)$,

- $p(y) = w_1^y + w_2^y + w_3^y + d(l_1^y, l_2^y) + d(l_2^y, l_3^y)$, *where* $Y \subseteq N$ *is the set of rejected orders, and*
- $t_l(k) = \sum_{i=1}^{3m^k} \max(0, (e^n_{i \bmod 3+1} - t_i^k))$, *where* $n \in N$ *is the order corresponding to location* l_i^k *in* p^k.

The static 1-PDRPTW is very similar to the static Pickup-and-Delivery Problem with Time Windows, and it can be solved by any algorithm that solves the PDPTW. To make this problem suitable for testing the dynamic behaviour of planning algorithms, we have to add uncertainties to the problem.

3.2 The Static Problem Augmented with Uncertainties

There is a long list of different event types that can influence logistical operations. There are some that add or remove elements from the given problem. For example, such events include new order arrivals, cancellation of orders, and the addition or removal of trucks. Other event types change certain parameters of the problem. In principle, all of the parameters may change. The most common events are delays at customers, which extend service times, and delays on the road, which change travel times. Additional event types include changes in orders, communication problems between drivers and planners, etc. From all possible event types, this thesis discusses the following three:

1. new order arrivals, which represent *release-time uncertainty*,
2. truck removals, which represent *truck-breakdown uncertainty*, and
3. delays at customers, which represent *service-time uncertainty*.

The event types represent different *uncertainty categories*, each of which describes a single source of uncertainty that can be present in a problem. We denote uncertainty categories by capital letters (R, S, B). Within the categories, different sub-categories, which we call scenarios are distinguished by a lower index that provides information on the events in the instances of the scenario (e.g. R_0 or R_{100}, where the lower index shows the percentage of dynamically released orders in the instances).

The main idea behind defining uncertainty categories separately from the transportation problem is that this way we can extend any transportation problem with any uncertainty or any combination of uncertainty categories. We introduce an upper-index notation to denote the uncertainty type that augments any given transportation problem. We designate TP^U to denote the transportation problem, TP, augmented by uncertainty category U. E.g. VRP^R denotes the Vehicle Routing Problem with Release-Time Uncertainty. The value of augmenting transportation problems with uncertainties is that it puts planning into the time perspective. By revealing the full problem to the planning method gradually, the nature of the problem changes fundamentally as the time allowed for computations is implicitly limited in such problem instances. Essentially, planning changes from the straight-forward action of receiving an input problem instance and

producing an output solution, to a feedback loop. In a problem with uncertainty, the input of the planning method is still a static transportation problem instance, but this instance is not fixed – it is modified by the previous output of the planning method, the plans, and the events that happen during execution of the plans.

In addition to the issue of information being gradually revealed, planning methods for transportation problems with uncertainties face another difficulty. Such problems require a planning method to coordinate the movements of vehicles in order to minimize the transportation effort regarding certain measures (e.g. empty distance). The transportation effort is not only the direct outcome of the plans produced by the method, but also the result of *executing the plans* of the planning method in an uncertain environment. The consequence is that the planning method cannot directly compute the optimal solution to the full problem; it only has an influence on the results by means of the plans it computes. The plans produced by the planning method and the dynamic events happening during execution together define the required effort spent on solving the problem.

If a planning method could be informed by an oracle about the future events, it could, in principle, produce plans that need no updates after every event. This would be similar to an end-of-the-day analysis, in which we try to find the best possible tours knowing what has happened during the day. To compute such *a posteriori* optimal plans, we have to provide the planning method with the information on the uncertainties *beforehand*, in order to allow it to consider all uncertainties. Planning methods that try to solve Transportation Problems with Uncertainties without such clairvoyance can only take the dynamic events into account *after* they happened. In the following subsections we provide further details on the three uncertainty categories and their scenarios.

3.2.1 Release-Time Uncertainty

New order arrivals are the most commonly considered source of uncertainty in vehicle routing problems. In contrast to the a posteriori case, in which all orders are revealed to the planning algorithm at the start of planning, new orders arrive throughout the day; as such the planning method must re-plan. This covers several different real-life problems, and therefore it is an interesting case to study.

In our release-time uncertainty instances, we differentiate between *static* and *dynamic* orders. Static orders are available for transportation (and known to the system) at the start of the planning horizon. They might have time windows that allow only later servicing, but the solution algorithm can consider these orders in the plans already from the beginning. In contrast, dynamic orders are released (thus revealed to the system) throughout the day, shortly before their execution must begin in order to meet the time windows.

To define different dynamic cases, we create scenarios that contain different percentages of dynamic orders. Scenario R_0 denotes problem instances without dynamic orders. This is also called the static scenario – a scenario in which all orders are known at the start of the day. The other extreme is R_{100}, which contains only dynamic orders. In general, scenario R_x denotes instances with an x percentage of dynamic orders.

The following definition augments the static 1-PDRPTW with job release-time uncertainty. This is done by associating release times to orders. Release times might refer to the start of the

planning (e.g. time 0), in which case the corresponding order is a static order. Release times that represent later time points define dynamic orders. As discussed above, assigning release times to orders (just as augmenting the problem with any other type of uncertainty) introduces a new dimension in the problem: time. Timing is relevant not only for the actual transportation of the orders, but also for the planning. Here planning is also an act in time, and plans that are computed before the release time of an order, cannot consider that order.

Definition 3.2

Name: *the Truckload Pickup-Delivery-and-Return Problem with Time Windows and Release-Time Uncertainty (1-PDRPTW^R).*

Instance:

- *the same as in the 1-PDRPTW in Definition 3.1, plus*
- *a sequence t_n^R of release times $\forall n \in N$, such that order n and t_n^R itself is not known to the planning algorithm during execution until t_n^R.*

Solution: *the same as in the 1-PDRPTW.*

Measure: *the same as in the 1-PDRPTW.*

Definition 3.2 augments the 1-PDRPTW by release times that disclose orders in the problem gradually. In the following section, we augment the same static problem with a different uncertainty category, which nevertheless also extends the orders with a different property.

3.2.2 Service-Time Uncertainty

When trucks execute their plans, they travel to locations to service customers. Operations at each location take a certain amount of time. Human planners learn typical service times by experience and use these values in planning. In our container-transportation test case, for example, the planners experience indicates that the pick-up of a container at the container terminal (including all paperwork) will take one hour; loading or unloading at the customer also takes one hour; return time at the sea terminal takes half an hour on average.

In reality, actual handling times may vary slightly. There are many reasons for this variation such as the availability of personnel or equipment, problems with the papers at customs, or some delay due to the customer. When service times in execution are different from the planned service times, plans may be invalidated. Planning systems should follow the execution of plans, and apply changes when necessary.

Service-time uncertainty scenarios are defined by adding actual service times to orders in the instances. Scenario S_X denotes problem instances for which the realized service time values have a standard deviation of X seconds. For example in S_{600}, actual service times vary by $\pm$ ten minutes (600 seconds).

Definition 3.3 introduces a new version of 1-PDRPTW augmented with service-time uncertainty. It extends the description of each order by three time durations, one for each location

of the corresponding order. These three new durations represent the actual service times at the corresponding locations. The planning methods, however, should not consider these actual durations until they are realized. The plan should be built using the original time durations inherited from the static problem.

Definition 3.3

Name: *the Truckload Pickup-Delivery-and-Return Problem with Time Windows and Service-Time Uncertainty (1-PDRPTW^S).*

Instance:

- *the same as in the 1-PDRPTW in Definition 3.1, plus*
- *a set* $\{w'^n_1, w'^n_2, w'^n_1\}$ *of actual service/waiting times* $\forall n \in N$ *replacing* $\{w^n_1, w^n_2, w^n_3\}$*, such that the actual times are not known to the planning algorithm until the corresponding service of the order is finished.*

Solution: *the same as in the 1-PDRPTW.*

Measure: *the same as in the 1-PDRPTW.*

The introduction of actual service times to the problem does not directly change the orders in the input to the planning method. It does, however, change the state of the trucks assigned to these orders. The planning method perceives the actual service times only through the unexpected states of the trucks. The next uncertainty category, truck-breakdown uncertainty, is more similar to release-time uncertainty in that the number of trucks in the input of the planning method changes at certain times.

3.2.3 Truck-Breakdown Uncertainty

When a truck breaks down for the rest of the day, planning algorithms are forced to reconsider the problem with one less truck. A breakdown event influences only one truck (or one route in case of the centralized planner), which forces the reallocation of all the unserviced orders of the given truck. Such a breakdown event my be seen as the simultaneous release of multiple new orders that must be served with one truck less in the fleet.

We form categories of truck-breakdown events by denoting the number of trucks that breaks down during the day. Category B_x represents instances that contain x truck breakdowns distributed (e.g. uniformly) over a day. For example an instance in category B_3 contains exactly three truck-breakdown event.

Definition 3.4 extends the static 1-PDRPTW with truck breakdowns, by specifying a breakdown time for trucks. In the instances, a special value (e.g. 0) is assigned to trucks that do not break down, and a different value to trucks that should break down.

Definition 3.4

Name: *the Truckload Pickup-Delivery-and-Return Problem with Time Windows and Truck-Breakdown Uncertainty (1-PDRPTW^B).*

Instance:

- *the same as in the 1-PDRPTW in Definition 3.1, plus*
- *a sequence t_k^B of breakdown times $\forall k \in K$, such that these times are not known to the planning algorithm during execution.*

Solution: *the same as in the 1-PDRPTW.*

Measure: *the same as in the 1-PDRPTW.*

New orders, truck breakdowns, or variable service times are all individual sources of uncertainty in a transportation problem. The above defined problems augment the 1-PDRPTW with only a single source of uncertainty. In the next section, we introduce a problem where multiple sources of uncertainties are present.

3.2.4 Mixed Sources of Uncertainty

From the *pure* uncertainty categories introduced in previous sections, we can generate *mixed* categories that represent mixed sources of uncertainty. Augmenting a problem with a mixed uncertainty category consists of simultaneously extending the problem with multiple independent uncertainty categories. Since the uncertainty categories we have defined above are independent in the sense that they add different properties to the problem in a non-contradicting way, such an aggregation is possible. It is expected that other uncertainty categories that might be defined later will also be independent from each other and from these uncertainty categories. This expectation is based on the idea that different uncertainty categories represent different, independent sources of uncertainty.

Following the above reasoning, scenarios of mixed uncertainty categories can be described as a Cartesian product of scenarios of the corresponding uncertainty categories. For example, the mixed scenario $R_{50} \times B_3$ refers to a set of problem instances in which half of the orders are released during execution, and three trucks break down while on the road. Out of length considerations, we use the shorter notation of $R_{50}B_3$ to denote such mixed scenarios.

Definition 3.5 augments the 1-PDRPTW with three independent sources of uncertainty: release-time, service-time, and truck-breakdown uncertainties. According to what has been described before, instances of this problem have a certain percentage of dynamic orders, the actual service times at the three locations of orders are different than the original service times, and some of the trucks break down during the day.

Definition 3.5

Name: *the Truckload Pickup-Delivery-and-Return Problem with Time Windows, Release-Time, Service-Time, and Truck-Breakdown Uncertainty (1-PDRPTW^{RSB}).*

Instance:

- *the same as in the 1-PDRPTW in Definition 3.1, plus*
- *a sequence t_n^R of release times $\forall n \in N$, such that order n and t_n^R itself is not known to the planning algorithm during execution until t_n^R.*
- *a set $\{w_1'^n, w_2'^n, w_1'^n\}$ of actual waiting times $\forall n \in N$ replacing $\{w_1^n, w_2^n, w_3^n\}$, such that the actual waiting times are not known to the planning algorithm until the corresponding service of the order is finished.*
- *a sequence t_k^B of breakdown times $\forall k \in K$, such that these times are not known to the planning algorithm during execution.*

Solution: *the same as in the 1-PDRPTW.*

Measure: *the same as in the 1-PDRPTW.*

Having all three sources of uncertainty in this problem yields a possibly higher level of uncertainty in a given instance. It can happen that the actual input of the planning method is more greatly influenced by the events (via the feedback loop) than by the original static problem instance. In such circumstances, we expect that it is less important to find an optimal solution for the static problem, and it is more important to be able to accommodate changes.

In this thesis we use the family of 1-PDRPTW^U to compare a distributed agent-based and a centralized optimization-based planning algorithm. The two algorithms are based on different representations of this problem, which are discussed in Section 3.3. We expect that some sources of uncertainties will be handled by one of the methods better than the other, and similarly, for any of the methods, some uncertainty types will be more difficult than others. To be able to quantify how well a planning method handles certain uncertainties, we define a *robustness* measure in Section 3.4.

3.3 Two Formulations of the 1-PDRPTW^U

The Truckload Pickup-Delivery-and-Return Problem with Time Windows and Uncertainties is composed of orders that should be transported between locations, and vehicles that can transport orders. The solution to the problem is a set of tours, and a set of rejected orders. As described in Chapter 2, such problems can be solved in two distinct ways: centralized optimization, or distributed agents. In order to facilitate the description of the two solution approaches in Chapter 4, we formulate this problem both using agents, and mixed-integer programming (MIP) in the following section. In both cases, we define the model of the static 1-PDRPTW first, and then discuss the extensions needed to model the 1-PDRPTW^U.

3.3.1 An Agent-based Formulation

The Truckload Pickup-Delivery-and-Return Problem with Time Windows and Uncertainties consists of vehicles and orders, just as any other vehicle routing problem. Therefore an agent

representation of this problem is built around two basic agent types: vehicle and order agents. Some representations in the literature go beyond these simple agent types, and introduce further agents, such as the transportation company, or customer agents. In the approach of Mes et al. (2007) for example, company agents coordinate the actions of a group of vehicle agents, while customer agents supervise a set of order agents. Such a hierarchical setup can, in principle, enhance the overall quality of the solutions via centralized decision making. Our goal, however, is to approve or disapprove the hypothesis that decentralized decision making techniques provide more robust planning. Therefore we do not include any such hierarchical agents capable of making centralized decision; on the contrary, we keep the architecture totally distributed.

The solution of a vehicle routing problem with or without uncertainty (also of the 1-PDRPTW^U) is found by assigning orders to vehicles. The assignments can be represented by contracts, and the whole transportation problem is usually defined as a contract network. The following sections introduce the three building blocks of the contract-network representation for the 1-PDRPTW^U: the contracts, the order agents, and the vehicle agents. This model can be solved in a variety of ways by algorithms designed for distributed decision making: algorithms where each agent computes its own decisions. Our solution method is discussed later in Chapter 4.

Contracts

A contract represents an agreement between an order and a vehicle agent. Both agents agree that the vehicle agent organizes transportation for the order agent according to the given requirements. Contracts are continuously made and destroyed by the agents during the solution of the problem. The contracts that are in effect at the end of the day (called the set of *final contracts*) are the ones that define which orders are transported by which vehicles, and which orders are rejected. Although the set of final contracts is the output of the planning method, it does not directly represent a solution to the 1-PDRPTW. The final solution is defined as the tour traveled by the vehicles, and which is not only influenced by the contract, but also by the events defined in the problem.

Definition 3.6 defines contracts as a pair of parties (agents of strictly different types), and two sets of requirements, one for each party. Required information from the order agent consists of the description of the locations that have to be visited and the time windows within which they have to be visited. Required information from the vehicle agent consists of the price that needs to be paid for the transportation service.

Definition 3.6 (Contract)

Properties

- *a pair of agents* (n,k), $n \in N$, *and* $k \in K$,
- *a list of location and time-window pairs* $(l_i^n, [s_i^n, e_i^n]), i \in \{1,2,3\}$ *as the required locations to visit for order n within the given time windows*
- *the price pr required by vehicle k.*

This contract structure defines the basic structure of the contract network by specifying which agents are involved in an agreement, and what conditions they might lay down. The following sections define the two agent types that actively participate in making such contracts.

Order Agents

Agents in this and the following sections are defined by their properties, their goals, and the constraints they must obey. Definition 3.7 describes order agents according to this methodology.

Order agents represent transportation requests in the problem, therefore their properties describe such requests. Accordingly, the basic properties of order agents are the *list of locations* that must be visited in order to service the order, the *service times* for each location, and the *time windows* for each location within which the servicing of the given order at the given location should commence.

The goal of order agents is to find a contract that minimizes the total price paid for transport by all order agents as a collective. This global goal can be translated into the local goal of requiring order agents to find contracts with a minimal price, but to also accept contracts with a higher price if it allows another order agent to make a contract at a better price, and the benefit to the other agent is more than the loss of the first agent.

To avoid contracts that would yield a longer empty travel than the rejection penalty, as defined in Definition 3.1, a constraint is added to the definition of order agents. This constraint requires the agents to only accept contracts with a price lower than 2 times their loaded time. The loaded time of an order will invariably be part of any price proposal it receives, since any truck will have to spend at least the loaded time on the transportation of the order. Anything above the loaded time will be spent on empty travel. That is, when the best option an order agent can find is worse than twice its rejection penalty, that means that the winning truck would spent more time traveling empty than the loaded time of the order, which is its rejection penalty, therefore such a contract should not be accepted.

Definition 3.7 (Order Agent)

Properties

- *a list* (l_1, l_2, l_3) *of three locations,* $l_i \in L$,
- *a list* (w_1, w_2, w_3) *of three waiting times,*
- *a list* $([s_1, e_1], [s_2, e_2], [s_3, e_3])$ *if three time windows,*
- *a contract* c.

Extended Properties

- t^R*, denoting the release time of the order,*
- (w'_1, w'_2, w'_3)*, denoting the actual waiting times at the three locations of the order.*

Goals

- *find a contract* $c' \in C^F$ *which minimizes* $\sum_{c \in C^F} pr_c$*, where* C^F *is the set of final contracts.*

Constraints

- *the accepted pr_c price cannot be larger than twice the loaded distance of the order (expressed in time):*

$$pr_c \leq 2(w_1 + d(l_1, l_2) + w_2 + d(l_2, l_3) + w_3).$$

When the 1-PDRPTW is extended with uncertainties, order agents have additional properties (see the extended properties in Definition 3.7). The release time t^R of an order is the time when the agent starts its activities. This way orders do not enter the tours before their release time. The w'_i actual waiting times redefine the original w_i waiting times, turning them into an estimation. Agents should consider the original waiting times during planning, but they should observe the redefined times during execution.

Order agents as defined above represent transportation requests in a problem. If the problem contains uncertainty, then the extended properties of the agents describe how these should be dealt with. The last element of the agent-based representation of the 1-PDRPTW^U, the vehicle agent, is defined in the next section.

Vehicle Agents

A vehicle in the 1-PDRPTW is represented by a vehicle agent. The basic properties of a vehicle agent are its *home location*, its *current location*, the *set of contracts* it has made with order agents, and its *plan*, which is a path consisting of the locations of the contracted orders.

The home location is where the vehicle should start and finish its operations. The current location is periodically updated to represent the location of a real vehicle serving the orders assigned to the vehicle agent. To coordinate their activities, vehicle agents make plans. A plan of an individual vehicle agent is defined by the orders with which the agent has a contract. The plan is described as a series of location-time pairs prescribing the order in which the locations of the contracted orders must be visited, and associating a time point with each visit.

The constraints defined on the plan of a vehicle agent express two limitations. First, the locations of the plan must be a concatenation of the locations of the contracted orders. This disallows the vehicle agent to move from a location of an order A to a location of another order B, unless it is moving from the last location of order A to the first location of order B. This means that a vehicle can transport only one order at a time. Second, the time points in the plan are restricted to the time windows of the corresponding order, meaning that the vehicle agent must obey the given time windows.

The goal of a vehicle agent is to maximize its profit, which is defined as the difference between the price it receives for transporting the orders and the cost of providing transportation. The total price a vehicle agent receives for transportation is defined by the prices in its contracts. The cost of transportation is defined as the time needed to travel along the path defined in the plan, plus the waiting times associated with the locations of the contracted orders.

Definition 3.8 (Vehicle Agent)

Properties

- *a home location* $l_h \in L$,
- *a current location* l,
- *a list of contracts* $C = (c_1, c_2, \ldots c_m)$, *and*
- *a path (plan)* p *as a list of location-time* (l_i, t_i) *pairs, where* $i = 0..3m+1$, *and*
  - $l_0 = l_{3m+1} = l_h$, *and*
  - $t_0 = 0$.

Extended Properties

- t^B, *denoting the time the vehicle breaks down.*

Goals

- *maximize profit (price - cost):*

$$\max \sum_{c \in C} pr_c - \sum_{i=0}^{3m} d(l_i, l_{i+1}) - \sum_{i=1}^{m} (w_1^n + w_2^n + w_3^n),$$

  where $l_i \in p$, *and* n *is the order with contract* $c_i \in C$.

Constraints

- *a vehicle can transport only one order at a time*
- p *should be such that* $\forall i \bmod 3 = 1 \wedge 0 < i < 3m+1$: n *is the order with contract* $c_{\lceil i/3 \rceil} \in C$, *and*
  - $l_i = l_1^n$, $t_i \in [s_1^n, e_1^n]$,
  - $l_{i+1} = l_2^n$, $t_{i+1} \in [s_2^n, e_2^n]$,
  - $l_{i+2} = l_3^n$, $t_{i+2} \in [s_3^n, e_3^n]$,

  and that $\forall i = 0..3m$
  - $t_{i+1} - t_i \geq d(l_{i+1}, l_i)$.

To represent vehicles in the 1-PDRPTW when it is augmented with truck-breakdown uncertainty, the vehicle agent is extended by a breakdown time (see the extended properties in Definition 3.8). The breakdown-time property specifies the time when the vehicle stops serving orders and releases all previously contracted orders.

It is interesting to point out, how the global objective of the original problem (cost minimization) is translated into local objectives of the agent-based formulation. Order agents' choice for a cheap transportation and truck agents' strive for profit maximization together lead to the global goal of cost minimization. Given that the MIP formulation described in the next section also defines a global objective function, it is reasonable to ask whether these very different formulations are equivalent to the original problem or not. They surely define the same set of

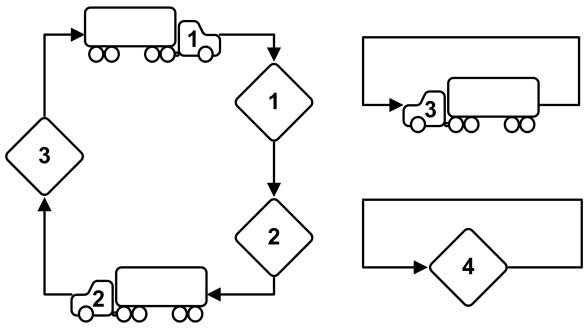


Figure 3.1: Cycles in the MIP solution structure.

feasible solutions, since they have been formulated to do so, but they naturally have different objective functions. While a formal proof of this is not given here, the two alternatives have been formulated with the intention that they represent the same problem, the problem described in previous sections.

3.3.2 A MIP Formulation

The MIP representation of the 1-PDRPTW is an adaptation by Jordan F. Srour (Máhr et al., 2010) of the mixed-integer program described by Yang et al. (1999). This is a model of a Pickup-and-Delivery Problem with Time Windows, and it was selected because of its ability to provide competitive results in comparison to fast running heuristics (Yang et al., 2004).

Before introducing the notation and mathematical formulation for this MIP, we begin with a small example to illustrate exactly how Yang et al.'s MIP works to exploit the structure of a truckload pick-up and delivery problem with time windows. The problem is modeled as a network, where the nodes represent vehicles and orders. Imagine a scenario with a network of seven nodes: three vehicles and four jobs. The model of Yang et al. is constructed such that it will find a set of least cost cycles describing the order in which each vehicle should serve the jobs. Figure 3.1 shows an example outcome, which is a tour from vehicle 1 to job 1, then job 2, then vehicle 2, then job 3, then back to vehicle 1. This would indicate that vehicle 1 serves job 1 and 2, while vehicle 2 serves job 3. The cycle including only vehicle 3 indicates that vehicle 3 remains idle. Similarly, the cycle including only job 4 indicates that job 4 is rejected. Whenever a solution to this MIP is found, it represents an execution plan for the trucks and orders present in the problem.

To be able to define the MIP for the 1-PDRPTW, we designate the following notation for the given information.

K the total number of vehicles available in the fleet.
N the total number of orders.
d_{ij} the travel time required to go from order i's last location to the first location of order j. Note, if $i = j$ then the travel time d_{ii} represents the loaded distance of order i; this distance includes the time from pick up at the first location to completion of service at the last location.
d^k_{0i} the travel time required to move from the location where truck k started to the first location of order i.
d^k_{iH} the travel time from the last location of order i to the home location of vehicle k.
v^k the time vehicle k becomes available.
τ_i^- earliest possible arrival at order i's first location.
τ_i^+ latest possible arrival at order i's first location.
M a large number set to be $2 \cdot \max_{i,j}\{d_{ij}\}$.

Note: τ_i^- and τ_i^+ are calculated to ensure that all subsequent time windows (at subsequent locations of the order) are respected. That is

$$\tau_i^- = max(s_1^i, s_2^i - d(l_1^i, l_2^i), s_3^i - d(l_2^i, l_3^i) - d(l_1^i, l_2^i)),$$

and

$$\tau_i^+ = min\left(e_1^i, e_2^i - d(l_1^i, l_2^i), e_3^i - d(l_2^i, l_3^i) - d(l_1^i, l_2^i)\right).$$

Given the problem of interest, we specify the following two variables.

x_{uv} a binary variable indicating whether arc (u, v) is used in the final routing; $u, v = 1, \ldots, K+N$.
δ_i a continuous variable designating the time of arrival at the first location of order i.

Using the notation described above, we formulate the MIP of the 1-PDRPTW that explicitly permits, although penalizes, job rejections, based on the loaded distance of a job.

$$\begin{aligned} \min \quad & \sum_{k=1}^{K}\sum_{i=1}^{N} d^k_{0i} x_{k,K+i} + \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij} x_{K+i,K+j} \\ & + \sum_{i=1}^{N}\sum_{k=1}^{K} d^k_{iH} x_{K+i,k} \end{aligned} \tag{3.1}$$

such that

$$\sum_{v=1}^{K+N} x_{uv} = 1 \qquad \forall u = 1,\ldots,K+N \tag{3.2}$$

$$\sum_{v=1}^{K+N} x_{vu} = 1 \qquad \forall u = 1,\ldots,K+N \tag{3.3}$$

$$\delta_i - \sum_{k=1}^{K} (d_{0i}^k + v^k) x_{k,K+i} \geq 0 \qquad \forall i = 1,\ldots,N \tag{3.4}$$

$$\begin{aligned} &\delta_j - \delta_i - M x_{K+i,K+j} + \\ &\quad (d_{ii} + d_{ij}) x_{K+i,K+i} \\ &\quad \geq d_{ii} + d_{ij} - M \end{aligned} \qquad \forall i,j = 1,\ldots,N \tag{3.5}$$

$$\tau_i^- \leq \delta_i \leq \tau_i^+ \qquad \forall i = 1,\ldots,N \tag{3.6}$$

$$\delta_i \in \mathbb{R}^+ \qquad \forall i = 1,\ldots,N \tag{3.7}$$

$$x_{uv} \in \{0,1\} \qquad \forall u,v = 1,\ldots,K+N \tag{3.8}$$

In words, the objective (3.1) of this model is to minimize the total amount of time spent traveling without a profit generating load. Specifically, we wish to minimize the penalty incurred from rejecting jobs, and the time spent traveling empty to pick up a container, between containers and when returning to the home depot. This objective is subject to the following seven constraints:

(3.2) Each order and vehicle node must have one and only one arc entering.

(3.3) Each order and vehicle node must have one and only one arc leaving.

(3.4) If order i is the first order assigned to vehicle k, then the start time of order i (δ_i) must be later than the available time of vehicle k plus the time required to travel from the available location of vehicle k to the first location of order i.

(3.5) If order i is followed by order j then the start time of order j must be later than the start time of order i plus the time required to serve order i plus the time required to travel between order i and order j; if however, order i is rejected, then the pick up time for job i is unconstrained.

(3.6) The arrival time at the first location of order i must be within the specified time windows.

(3.7) δ_i is a positive real number.

(3.8) x_{uv} is binary.

This model specification serves to find the least-cost (in terms of time) set of cycles that includes all nodes given in the set $\{1,\ldots,K,K+1,\ldots,K+N\}$. We define $x_{uv},(u,v=1,\ldots,K+N)$ to indicate whether arc (u,v) is selected in one of the cycles. These tours require interpretation in terms of vehicle routing. This is done by noting that node $k,(1 \leq k \leq K)$ represents the vehicle k and node $K+i,(1 \leq i \leq N)$ corresponds to order i. Thus, each tour that is formed may be seen as a sequential assignment of orders to vehicles respecting time-window constraints.

The above described model can be solved to find the solution for the static 1-PDRPTW, but the same model can also be applied in an on-line manner to problems with uncertainty. In such cases, the current state of the world is periodically encoded using this MIP, which is then solved to find an optimal assignment of orders to vehicles at each iteration. That is, the adaptation of this model to 1-PDRPTW^U does not happen by adjusting the model itself, but by applying it iteratively, and by considering the uncertain events during the encoding of the actual state of the world in the next MIP. This approach is described in detail in Chapter 4.

Thus far in this chapter we defined the main sources of uncertainty that are studied in this thesis; we defined the transportation problem that is used to compare a distributed and a centralized planning method; and we provided two separate models of this problem: one that suits a distributed approach and another one that suits a centralized approach. In the following, we define the measures that are used to judge the performance of the competing methods.

3.4 Robustness of a Routing Method

In our comparative experiments of transportation planning methods, we evaluate the methods following a dual approach. We evaluate the methods not only by using a traditional cost-based measure, but also by comparing their robustness using a measure that is defined in this section.

One of the possible interpretations of robustness in a computer system is that it is the property of handling exceptional inputs well. Exceptional inputs of a logistical planning system are the uncertain events encompassing changes in orders, truck breakdowns, or traffic jams. Whether a planning method handles such events well can be measured by comparing the execution of transportation plans produced by the given planning method to the *a posteriori* optimal execution. While such a comparison is undoubtedly useful, unfortunately it can unfortunately sometimes be quite difficult to compute an a posteriori optimum. Instead of using the optimal solution, we base our robustness measure on the comparison of the performance of a method in static versus dynamic instances. This quantifies the degradation of performance between the static and uncertain instances, and serves as an expression of how well a method handles the uncertainties present in the latter cases.

As a start, let us remember that Defintion 3.1 defines the cost of transportation as the sum of the empty travel times incurred by the vehicles, plus the sum of the rejection penalty of orders not transported, plus the sum of the lateness for orders that were delivered after their time windows. Let us denote this cost of transportation for a method in instance i as $c(i)$, and the average cost for a scenario S as

$$c(S) = \frac{1}{|S|} \sum_{i \in S} c(i).$$

While this cost measure is defined for any scenario, static (R_0) or dynamic (e.g. R_{100} or B_{10}), the following robustness measure is only defined for dynamic scenarios. It expresses the degradation of solution quality that a given planning method suffers due to the uncertainty present in the instances of scenario S. To appreciate this measure, note that any dynamic instance i was generated from a static instance $\hat{\imath}$ by adding uncertainty as defined by scenario S. Accordingly, we define robustness as follows.

Definition 3.9 (Robustness) *The robustness of a given planning method, for an instance i, is defined as the ratio of its total cost for instance $\hat{\imath}$ in the static scenario R_0 and the total cost of the corresponding instance i in the dynamic scenario S:*

$$r(i) = \min\left(1, \frac{c(\hat{\imath})}{c(i)}\right)$$

The average robustness on a set of instances S is then denoted by $r(S)$.

$$r(S) = \frac{1}{|S|}\sum_{i \in S} \min\left(1, \frac{c(\hat{\imath})}{c(i)}\right)$$

In this measure, the min() function ensures that if the cost of a method for a dynamic instance is lower than in the corresponding static instance, its robustness for that instance is maximized at 1. A robustness value of 1 means that the performance in the dynamic instance is at least as good as in the static instance (i.e. there is no performance loss).

These two measures, $c()$ and $r()$ describe two different properties of a method. The cost measure, $c()$, reveals how efficient the method is on the instances of a scenario. The robustness measure, $r()$, tells us how fragile this efficiency is. It defines the (normalized) reduction in performance under uncertainty. Intuitively, $c()$ and $r()$ are not independent, which is illustrated by the following proposition.

Proposition 3.1 *Given a dynamic instance i and the related static instance $\hat{\imath}$, if method B is more robust than method A on i, then the total costs of B for i are less than A, or the total costs of A for $\hat{\imath}$ are less than B (or both).*

PROOF To arrive at a contradiction, assume that B is more robust than A on i, the costs of B for i are at least as high as A, and the costs of A for $\hat{\imath}$ are at least as high as B. Denote the costs of method X on instance k by $c^X(k)$. This assumption yields that

$$\min\left(\frac{c^B(\hat{\imath})}{c^B(i)}, 1\right) > \min\left(\frac{c^A(\hat{\imath})}{c^A(i)}, 1\right) \tag{3.9}$$

$$c^B(i) \geq c^A(i), \text{and} \tag{3.10}$$

$$c^A(\hat{\imath}) \geq c^B(\hat{\imath}). \tag{3.11}$$

Dividing the left hand side of 3.11 by the right hand side of (3.10), and the right hand side of (3.11) by the left hand side of (3.10), we get that

$$\frac{c^A(\hat{\imath})}{c^A(i)} \geq \frac{c^B(\hat{\imath})}{c^B(i)}. \tag{3.12}$$

There are two cases. If $c^B(\hat{\imath}) \geq c^B(i)$ then with (3.11) and (3.10) $c^A(\hat{\imath}) \geq c^A(i)$. Thus both methods have a robustness value of 1, contradicting that B is more robust than A.

If $c^B(\hat{\imath}) < c^B(i)$ then $\frac{c^B(\hat{\imath})}{c^B(i)} < 1$ and thus from (3.9) it follows that $\frac{c^B(\hat{\imath})}{c^B(i)} > \frac{c^A(\hat{\imath})}{c^A(i)}$. This contradicts with Equation (3.12). ■

Proposition 3.1 highlights the underlying dependency between the cost and the robustness measures for a certain subset of the cases. Table 3.1 provides a more general overview on how

Table 3.1: The dependence of robustness on how the cost changes between static and dynamic instances.

method A	**method B**	**robustness**
$c^A(\hat{\imath}) \geq c^A(i)$	$c^B(\hat{\imath}) \geq c^B(i)$	$r^A(i) = r^B(i) = 1$
$c^A(\hat{\imath}) \geq c^A(i)$	$c^B(\hat{\imath}) < c^B(i)$	$r^A(i) = 1 > r^B(i)$
$c^A(\hat{\imath}) < c^A(i)$	$c^B(\hat{\imath}) \geq c^B(i)$	$r^A(i) < r^B(i) = 1$
$c^A(\hat{\imath}) < c^A(i)$	$c^B(\hat{\imath}) < c^B(i)$	any

Table 3.2: The dependence of robustness on the relation of costs in cases with a higher cost for dynamic instances (i).

static instance $\hat{\imath}$	**uncertain instance i**	**robustness**
$c^A(\hat{\imath}) = c^B(\hat{\imath})$	$c^A(i) = c^B(i)$	$r^A(i) = r^B(i)$
$c^A(\hat{\imath}) < c^B(\hat{\imath})$	$c^A(i) < c^B(i)$	any
$c^A(\hat{\imath}) \leq c^B(\hat{\imath})$	$c^A(i) > c^B(i)$	$r^A(i) < r^B(i)$
$c^A(\hat{\imath}) > c^B(\hat{\imath})$	$c^A(i) > c^B(i)$	any
$c^A(\hat{\imath}) > c^B(\hat{\imath})$	$c^A(i) \leq c^B(i)$	$r^A(i) > r^B(i)$

robustness and the cost measure are related. It reveals how changes in the transportation cost define which of the two methods are more robust. In the trivial case, when one of the methods produced a lower cost solution in the uncertain instance than in the static instance (e.g. its cost was lower in an R_{100} instance than in an R_0 one), and the other method did not, then the first method is more robust than the second. If both had a better performance in the uncertain case, then they are equally (and totally) robust. However, when both have a higher cost in the uncertain case than in the static case, there are four sub-cases to discuss.

Table 3.2 shows how robustness depends on the relation between the cost measures of the two methods applied to the *same* instances when both methods have a higher cost on a dynamic instance i, than on the related static instance $\hat{\imath}$. When method A and B have an equal performance in both i and $\hat{\imath}$ instances, then they are equally robust. When it is the same method that has less cost in instance i and in $\hat{\imath}$ (e.g. the on-line optimization has less cost both in R_0 and B_{10}), then any of the two methods can be the more robust. However, when Proposition 3.1 can be applied, because one of the methods has less cost in i and the other in $\hat{\imath}$ (e.g. the on-line optimization has less cost in R_0, but the agent method in R_{100}), then there is a clear winner. It follows from this discussion that the cost and robustness measures are not independent of each other. Simple rules can be defined to predict which method is more robust than the other based on the relation of the costs of the methods on static and dynamic instances. Still, it is not always enough to know which method has less cost on the given instances to tell which one is more robust. In such cases one also has to know the magnitude of cost differences in order to compute the robustness of the methods.

The measures provided in Definition 3.1 and 3.9 define quality and robustness of a method in a set of problem instances from the problem family of 1-PDRPTW^U. To analyse or compare

different planning methods, the measures can first be computed for a limited subset of the 1-PDRPTW^U, which contains only instances with a single uncertainty category (e.g. truck breakdowns). This way, robustness of a method *against* that single uncertainty type can be analyzed. Later, more complex scenarios can be tested by using a subset of 1-PDRPTW^U, which includes multiple uncertainty types. Then the computed measure expresses robustness against the given levels of the different uncertainty categories simultaneously present in the instances.

3.5 Summary

In this chapter, we studied the problem of formalizing the uncertainties in transportation problems, as raised by Research Question 1. The goal of the chapter was to introduce an uncertainty framework that can be used to extend static transportation problems with dynamic events. To this end, we first formalized the static Truckload Pickup-Delivery-and-Return Problem with Time Windows (1-PDRPTW). Then we introduced three uncertainty categories that add three different types of events to a transportation problem. For each type of uncertainty, we described how different scenarios can be defined within these uncertainty categories, thereby formalizing an extension of the static 1-PDRPTW. This approach also allows for the definition of a 1-PDRPTW with mixed uncertainties.

After defining the exact transportation problem of interest, we described two radically different representations of this problem. One of them, the agent-based representation, defines distributed decision making entities that make their choices independently of each other, based on limited information. The other one, the MIP representation encodes the state of the world (thus using full information), and computes locally optimal solutions iteratively. In addition to using full or partial information to make decisions, the two representations also differ in their sensitivity to changes in the problem. While a global MIP-based approach might produce a totally different solution when some part of the problem is changed, the partial-information-based decisions of the agents limit the effect of a single change in the problem.

Finally, as an answer to Research Question 1a, the uncertainty framework was completed by the definition of a robustness measure that quantifies the degradation of performance caused by uncertainties in the problem. First we defined a traditional cost-based measure, which can be measured in both static and uncertain instances. We based our robustness measure on the ratio of the cost measure in static and uncertain cases.

The uncertainty framework defined in this chapter, especially the transportation problem augmented by this framework, will be used in Chapter 5 to provide a basis for the simulation environment, and in Chapter 6 as the basis for the comparison of the planning methods. For the two representations of the 1-PDRPTW^U from Section 3.3, we present two solution approaches in the next chapter.

Chapter 4

Distributed and Centralized Approaches to VRPs with Uncertainty

4

A transportation problem with uncertainties (as defined in Chapter 3) is the problem of directing trucks to transport goods such that an optimal sequence of actions is performed, even in the presence of uncertainties. This work extends traditional transportation problem formulations with the introduction of uncertainty, and with the requirement of producing an optimal sequence of actions. To handle uncertainty, planning methods need to be modified to react to dynamic events. This chapter describes two such methods, a centralized and a distributed one. The description of these methods was published by Máhr et al. (2008, 2010).

The simplest way to handle uncertainty arising from events that change the current situation (number of trucks, number of orders, travel time, etc.) is to delay routing decisions until the last moment. Uncertainty tends to decrease over time, as more and more unknown factors become known. Larsen et al. (2002) discuss dispatching heuristics in the context of the Dynamic Traveling Repairman Problem, where methods do not make any plans beforehand, but direct trucks on-line to new jobs as they finish previous jobs. These naive approaches yield inefficient results due to a lack of coordination in the movements of the vehicles. Careful planning can coordinate execution, but planning methods are prone to uncertainties, since at the time of making plans, most of the uncertainties are still present in the problem.

One solution to this problem, applied in stochastic approaches (see Powell et al., 1995), is to accommodate the uncertainties by assuming a probability distribution of the possible outcomes, and producing plans that are optimal in expectation. This is an elegant way of using available information about events (truck breakdowns, late deliveries, etc.) that happened in the past, but it relies on the correctness and/or completeness of this information. There are deterministic approaches (as reviewed by Bianchi, 2000) that assume nothing about the possible events. Such methods produce plans ignoring the possibility that something can happen other than what was planned. When, however, such a deviation does happen, these reactive methods change the plans to adapt to the new situation. These deterministic reactive approaches are the focus of our investigation, because we argue that adapting to new situation is relevant even in the presence of a good stochastic solution.

Figure 4.1 depicts the partition of the space of solution methods to vehicle routing problems with uncertainties. The greyed-out parts contain our methods of interest, i.e. the distributed and centralized deterministic planning methods. To be able to handle uncertain problems, both of these approaches must be ready to react to uncertain events that occur during execution of the plans. The method of handling these events might be different in every approach, but still these approaches must adapt to new situations whenever they occur.

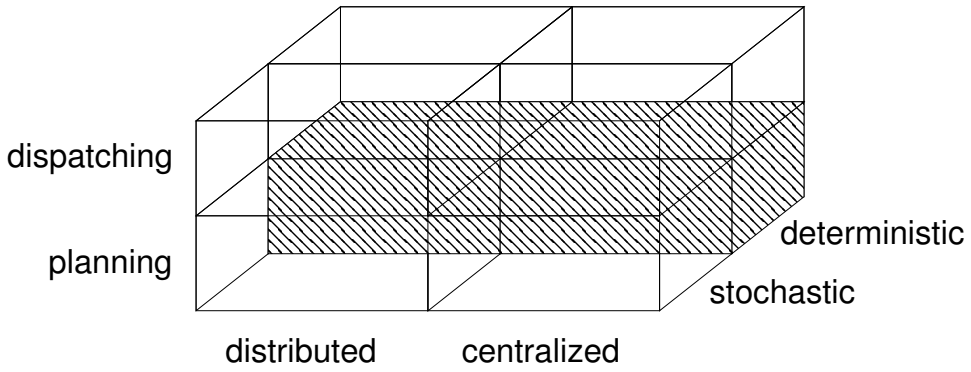


Figure 4.1: Categories of solution methods to VRPs with uncertainties.

Handling events and adapting to new situations can be done in a variety ways. Centralized methods use all available information to make decisions. In contrast, distributed methods base decisions only on locally available information, where localization of information is part of the model. If the problem size allows it, to make best use of all the concentrated information, centralized approaches calculate optimal plans. In contrast, since decentralized approaches do not have the advantage of access to full information, they employ fast heuristics to generate suboptimal solutions quickly. The interesting question is whether the flexibility provided by fast heuristics outweighs the optimal decisions based on complete information in situations for which the problem is burdened with uncertainties.

In order to address this question in this thesis, this chapter describes a distributed and a centralized deterministic planning approach. The distributed approach (in Section 4.1) employs the best fast heuristics in the literature, in addition to a new heuristic (the substitution algorithm) which has been developed to improve its performance. The centralized on-line optimization method (in Section 4.2) is an adaptation of the method of Yang et al. (1999) which has been shown to surpass simple (centralized) heuristics. A comparison of the two approaches is provided in Section 4.3 to analyze the fundamental differences.

4.1 Decentralized Multi-agent Approach

Multi-agent systems decentralize decision making by localizing information. The base of this localization is the agent model. When agents represent different real-world entities, communication between the real world and the agents becomes an integral part of the solution approach. We assume that agents have a permanent communication channel that connects them to their real-world counterparts, and that they use this channel to remain up-to-date about the state of the environment. This information forms the agents' *private knowledge base* that they use to make decisions.

In a transportation problem, trucks, orders, planners, transportation companies, or customers may be represented by agents. According to our previous assumption, the decisions of these agents are steered by the information they receive from the real entities they represent. Truck agents might be kept up-to-date about the location of their trucks, while planner agents could receive information to change their planning behavior.

In addition to the agents' private knowledge base, they may decide to share information with

one another. Shared information may improve the decisions that individual agents make, and therefore it may yield better overall results. How information sharing happens is, in part, specified by the agent model, and in part by the algorithms used to solve the model. For example, the agent model may contain agents with the specific role of aggregating information. An example is a planner agent that knows the plans of all its truck agents, and produces optimized plans centrally for those agents. In contrast to this central solution, agents may distribute information via a communication protocol. An example of this is presented as the container exchange protocol in Section 4.1.1. Being shared by other agents, or being part of an agent's private knowledge base, the information that is visible to an agent is what it bases its decisions on. The essence of agent-based distributed decision making is that agents have different information sources, therefore their knowledge base is different, and none of them has the 'whole picture'. As a result, their decisions will also differ in the end.

Communication between the real world and the agents is more important than simply keeping the information encapsulated by agents up-to-date. The news of changes received from the real world triggers the agents to reconsider their positions and to change the current solution. In a transportation problem, agents make plans for the coordination of transportation of orders. Whenever an unforeseen event happens, agents start sending messages, and using their local algorithms to adapt their plans to the new situation. Thus, instead of executing one systematic global search for the best adaptation, agents execute several local searches, and coordinate those searches by communication.

The basic elements of our agent model, such as vehicle (truck) agents, order agents, and contracts, were already introduced in Section 3.3.1. In the next section, Section 4.1.1, we describe the algorithms agents use to compute a solution.

4.1.1 Agent Algorithms

In building our agent system, our goal was to define a fully decentralized solution approach, in which no information, regarding different agents or different sub-problems, becomes concentrated at a higher level. Additionally, we wanted to equip our agents with state-of-the-art mechanisms to successfully solve the routing problem. In this section we describe the mechanisms our agents use to reach their goals.

Specifically, we first define the auctioning mechanism used to find the cheapest transportation option for a single order. Then, we introduce substitution as our extension to the simple insertion mechanism truck agents use to calculate their costs and bid on the auctions. Finally, we highlight the improvement techniques that both order agents and truck agents have at their disposal.

Auctioning Orders

Order agents organize auctions immediately after they enter the planning system. If orders are revealed to the system at well-spaced intervals, then the auctions are fully sequential. If, however, some orders are announced at the same time, or close to each other, those auctions are held in parallel.

Order agents collect quotes for transportation via these auctions. The bid that truck agent t sends in for the auction of order i contains the cost it would incur should it win the auction and

transport the order. This includes the loaded time of the order, which is the same for every truck, plus the extra costs of servicing the order, which is the empty travel each truck must undertake in order to service the order.

$$bid_t^i = d_{ii} + cost_t^i \tag{4.1}$$

Inequalities (4.2)-(4.4) show how such a bid structure leads to the rejection of orders that would cost more than their associated rejection penalty. An order agent i only accepts bids that are less than its reservation price (P_{res}^i). The reservation price is the sum of the loaded time and the rejection penalty of the order (thus twice the loaded time, see Section 3.3.1). This is done because the cost the truck incurs from transporting the order also includes the loaded time of the order. In this way, orders that would have a higher extra cost (empty travel) than their rejection penalty are rejected.

$$P_{res}^i > bid_t^i \tag{4.2}$$

$$2d_{ii} > d_{ii} + cost_t^i \tag{4.3}$$

$$d_{ii} > cost_t^i \tag{4.4}$$

Order agents implement a single-shot second-price closed-bid (a.k.a. Vickrey) auction. This auction type is popular in the literature because of its simplicity (Hoen and Poutré, 2003). We use this mechanism because by setting the price to the second-best bid, the market position of the order is implicitly communicated to the winning truck, and it is used to make substitution decisions. Having the second-best bid as the price also ensures that the truck agent realizes a profit. If the winner is the only truck agent bidding on the auction, or the second-best bid is higher than the reservation price, then the order agent sets the price of the contract to the reservation price.

The winning truck agent can accept the contract only if its plan is unchanged since the time of the bidding. If the plan has changed in the meantime, due to winning another order, for example, then the truck agent must re-calculate the transportation costs for this order considering the new plan. If the new costs are less than or equal to the price in the contract, the truck agent accepts the second order. Otherwise it rejects it, since transporting the order in this new situation costs more than the price it would receive for the job. If the winner rejects the order, the order agent will try to close a deal with the second, or the third, etc. truck agents in a similar manner. If all bidders decline the offered deal, the order agent remains unassigned.

When an order agent succeeds in making a contract with a truck agent, it sends a message to the other order agents to notify them about the changed state of the contracted truck agent. This information is crucial for unassigned orders, because they can send a new request-for-quote message to the given truck agent in the hope that they can make a deal now that the situation has changed. Every order agent has a latest possible auctioning time. After that time it is not possible to pickup and transport the order within the given time windows. Order agents try to re-auction themselves only if this latest possible auction time has not yet passed. To avoid an avalanche of auctions from rejected order agents every time a contract has been made, re-auctions take place at randomly chosen intervals with an exponential distribution and a mean value of one-tenth of the remaining time for the order to be successfully scheduled.

To establish the time and messaging complexity of the above described auction protocol, let

us first look at a single auction organized by an order agent. To clear an auction, the order agent sorts the bids of K truck agents. This can be done in $O(K \log K)$ time. In terms of messaging, to receive the bids, the order agent first has to send out K request-for-quote messages. The K replies to these messages contain the bids. After selecting the best bid, one message is sent to the winner containing a contract and one reply is received from the winner to acknowledge or decline the contract. In the end, one message is sent to all losing truck agents to notify them about not winning the auction. That is, in a single auction, at least $3K+1$ messages are sent. In the worst case, the order agent has to try to contract all truck agents corresponding to the ordered bids, which results in $4K$ messages in total.

After finding the time and message complexity of a single auction, in order to assess the total time and messaging complexity of the auctioning mechanism, we need to determine the worst-case number of auctions necessary to solve a vehicle routing problem. All order agents will hold a minimum of one auction, therefore the minimum number of auctions is N. This may be realized in a case when orders arrive distributed over the day, and no auctions are held parallel. Should they all have their first auction at the same time, however, only K contracts can be made in the worst case. Then, if the second round of $N-K$ auctions is synchronised again, it might also yield only K contracts. The sum of all such auctions in the worst case is $N+(N-K)+(N-2K)+\ldots$. The number of extra rounds needed after the first one is $m = \lfloor \frac{N}{K} \rfloor$. The total sum is $(m+1)N - \frac{m(m+1)}{2}K$, which is in the order of $(O(\frac{N^2}{K}))$.

Proposition 4.1 *The auction algorithm implemented by the order agents finds an allocation for all orders in $O(N^2 \log K)$, but in $\Omega(NK \log K)$ steps, using at most $O(N^2)$ and at least $\Omega(NK)$ messages.*

PROOF The time complexity of the algorithm is defined by sorting the N order agents to obtain a preferred list of the K truck agents during the auctions. Since every auction requires exactly one sorting, and the number of auctions in the worst case is $O(\frac{N^2}{K})$, the worst case time complexity of the algorithm is $O(\frac{N^2}{K}K \log K) = O(N^2 \log K)$. Since the minimum number of auctions is N, the algorithm needs at least $\Omega(NK \log K)$ steps.

Similar reasoning holds for the number of messages. We have shown above that an auction may require $4K$ messages in the worst case. Since there are $O(\frac{N^2}{K})$ auctions, the message complexity of the auctioning scheme is $O(N^2)$. In the best case, $3K+1$ messages are required for an auction, therefore the minimum number of messages required for N auctions is $\Omega(NK)$.■

In the following subsection, details of the algorithms used by the truck agents to calculate their bids are discussed.

Insertion with Substitution

When a truck agent bids on an order, its bid includes the loaded time of the order, and the additional cost it would incur should it win the auction and transport the order. To calculate this additional cost, a truck agent has to compute the difference between the cost of executing its plan both with the new order included and without it. In general, a truck agent needs to solve a TSP-like problem in order to find the optimal sequence of the orders in its plan. As discussed in Section 2.2.2, agent-based solutions typically do not try to find the optimal solution for such a

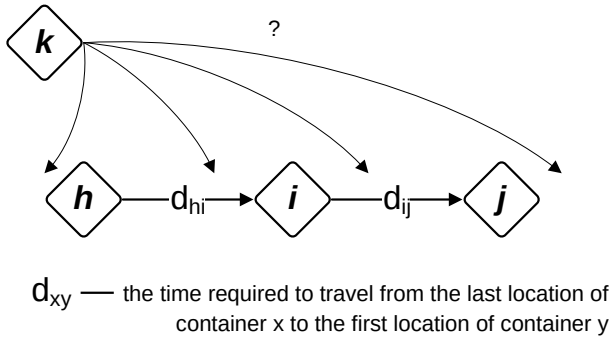


Figure 4.2: Positions considered by the simple insertion heuristic.

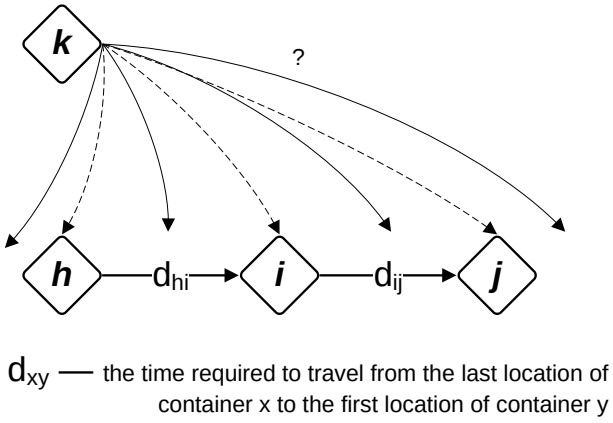


Figure 4.3: Positions considered by the extended insertion heuristic.

sub-problem; rather they use a sub-optimal but quick heuristic, the insertion heuristic. Our agent system is no different.

The insertion heuristic algorithm (as inspired by Solomon (1987)) searches through the space of solutions by inserting the new order into the plan at various locations between already contracted orders, and by appending the new order to the head or the tail of each plan (see Figure 4.2). The cost ins_{ij}^{k} of inserting order k between order i and j is calculated as the difference of the empty-travel times with and without order k:

$$ins_{ij}^{k} = d_{ik} + d_{kj} - d_{ij}, \tag{4.5}$$

where d_{xy} is the time the truck travels empty from the last location of order x to the first location of order y, if $x \neq y$. As noted earlier, when $x = y$, d_{xy} denotes the loaded time of the order. The algorithm finds the cheapest insertion by a simple linear search procedure considering subsequent insertion positions. This algorithm is quick, but the solution found can be arbitrarily far from the optimal. In addition to searching for the cheapest insertion, the trucks also check time-window constraints, and do not accept any solution that violates time windows. The bid that is submitted based on such an insertion is actually the sum of the insertion cost and the loaded time of the order.

We extended this basic algorithm to consider the effect of substituting already contracted orders. The technique of substituting old orders with new ones is motivated by the decommitment technique successfully applied in the literature to overcome the problem of making commitments

too early (Hoen and Poutré, 2003). The insertion heuristic is sensitive to the order of job arrivals. By allowing trucks to substitute earlier-committed orders, we reduce this ordering-sensitivity problem. In this extended version of the insertion heuristic, a truck agent not only computes the cost of inserting the new order between existing orders, but also computes the cost of delivering it *instead of* other orders (as seen on Figure 4.3). Just as in the case of the simple insertion algorithm, the computations are done in a single run over the plan. In step i of the extended algorithm, the cost of inserting order k before order i and the cost of substituting order i by order k is computed. The cost of insertion is the same as in the original algorithm. The cost of substituting order i, between order h and j in the plan, by order k is defined as:

$$subs_i^k = ins_{hj}^k + profit_{hj}^i. \tag{4.6}$$

The first term, inserting order k between h and j is as defined above. The lost profit of order i, $profit_{hj}^i$, is computed as the price offered for order i minus the loaded time and the insertion cost of order i.

$$profit_{hj}^i = price^i - d_{ii} - ins_{hj}^i. \tag{4.7}$$

The loaded time is subtracted because it is also part of the bids trucks submit, and therefore it is part of the price too. The insertion cost element of the lost profit is recalculated for every substitution decision, because the preceding or the subsequent order of i may have changed since it was originally included in the plan. Such a substitution cost supports the replacement of orders that had many similar competing bids. The price of such an order, computed from the second-best bid, is only a little bit higher than the associated costs. This means that the profit is low, therefore it becomes a good candidate for replacement. At the same time such an order can easily find another truck for a similar price, since there was at least one truck that bid very close to the winner. Along the same lines, it can be seen that the proposed calculation prevents the substitution of orders that could have difficulties in finding another truck for a similar price.

Similar to the insertion-only version, truck agents check time-window constraints and only accept an insertion or substitution if all time windows are respected. In case of a possible substitution, it is also checked whether the new order has a smaller insertion cost than the current one. An $\varepsilon \geq 0$ parameter of the algorithm defines the required distance between the new and the old insertion costs. The new order is accepted for substitution only if its insertion cost plus ε is less than the insertion cost of the current order. Algorithm 4.1 summarizes the extended insertion algorithm.

To define the complexity of this algorithm, we refer to the complexity analysis of the simple insertion algorithm in Solomon (1987). It takes a linear search to compute the insertion cost of every insertion position. To compute substitution costs together with the insertion costs, it only takes twice the number of steps, therefore the complexity is still $O(N)$. Then the algorithm orders the costs, which takes $O(N \log N)$ steps. Finally, to check the time constraints, the algorithm iterates over the costs one-by-one, and checks the whole plan for feasibility in case the given insertion or substitution is realized. This is done in $O(N^2)$. The following proposition summarizes the considerations above.

Proposition 4.2 *Algorithm 4.1 finds the best insertion or substitution location in a vehicle agent in* $O(N + N \log N + N^2) = O(N^2)$ *steps.*

Algorithm 4.1 Insertion and substitution of orders.

1. Iterate over the plan and collect the insertion and substitution costs corresponding to positions in the plan.
2. Sort the merged list of (insertion cost, position) and (substitution costs, position) pairs by increasing order of costs.
3. Iterate over the list of costs and positions, and
   (a) if the position indicates substitution (of container A to B) and the insertion cost of the new order B is not smaller by an $\varepsilon \geq 0$ than the insertion cost of the current order A ($\text{ins}_{XY}^{A} \leq \text{ins}_{XY}^{B} + \varepsilon$) then drop this alternative, and take the next cost-position pair,
   (b) if it is not possible to insert or substitute the new order at the given position without violating its time windows, then take the next alternative cost-position pair,
   (c) if the time windows of the subsequent orders are violated by the insertion or substitution of the new order, then take the next alternative, or
   (d) else return the position and the cost as the cheapest feasible insertion or substitution position.

PROOF The three components of the complexity in the proposition corresponds to the three steps of the algorithm. Step 1 is a linear search over the plan of the truck, it is $O(N)$. Step 2 orders the list of costs, it is $O(N \log N)$. Finally, Step 3 checks the time windows in the plan, possibly for all costs in $O(N^2)$ steps. ■

Algorithm 4.1 is used by truck agents to calculate their bids. The bid they submit is always the cheapest feasible cost found by Algorithm 4.1, plus the loaded time of the order. If a truck wins an auction where its bid corresponds to an insertion position, then it simply inserts the new order at the specified position. If the bid corresponds to a substitution position, then the truck first releases the order in that position and then inserts the new order in its place.

The released order starts a new auction just as it did following its arrival time. This new auction may, of course, result in another substitution, which invokes a new auction, and so on. We refer to such a sequence as a *substitution chain*. Figure 4.4 displays an example of such a chain ending in an insertion.

Definition 4.1 *A* substitution chain *is a sequence of related order substitutions in different trucks, where the next new order considered is always the one previously released in a substitution. The end of a substitution chain is either an insertion or a rejection.*

Long substitution chains occur in situations when, from round to round, substitution is the best alternative to incorporate the given order into the plans. We show that substitution chains never consist of infinite number of substitutions.

Proposition 4.3 *If truck agents use the insertion-substitution algorithm (Algorithm 4.1) to bid on auctions for orders, and substituted orders re-auction themselves after being released, then the resulting substitution chains are finite.*

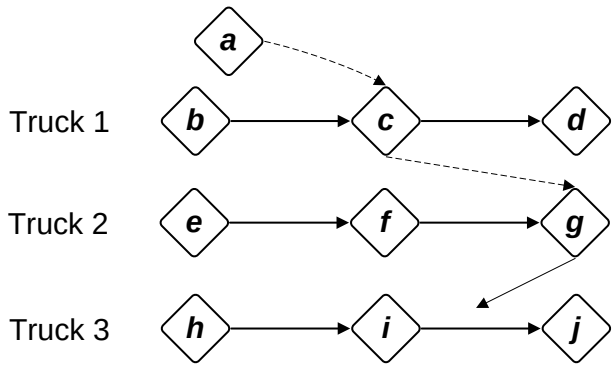


Figure 4.4: An example substitution chain ending in the insertion of order g between order i and j.

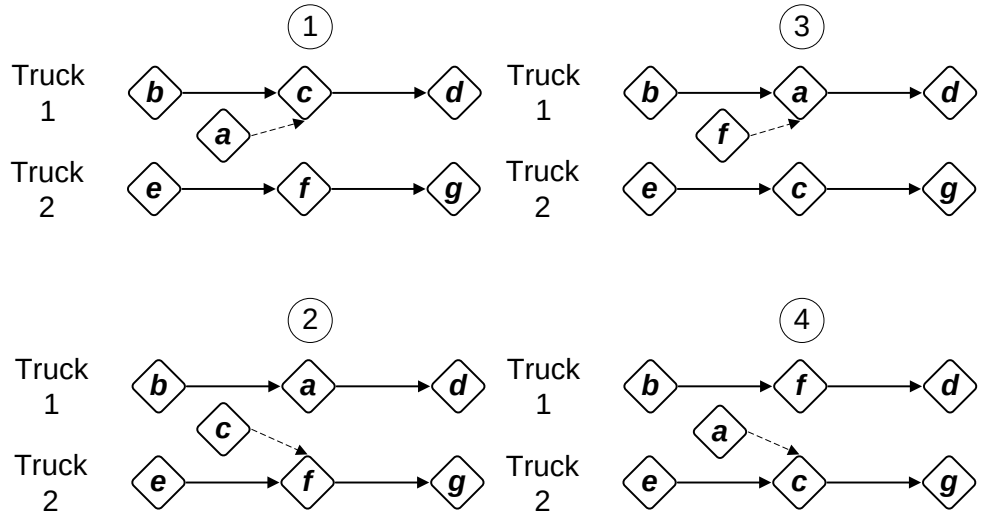


Figure 4.5: Four steps of a substitution chain where three orders substitute each other in a loop.

PROOF In the first part of the proof, we assume that there is only one substitution chain active at a time. This means that every change of any plan is a result of the currently active substitution chain. After this case, we discuss the interactions of simultaneously active substitution chains.

Since there are only a finite number of ways that orders can be incorporated into the plans of the trucks, an infinite substitution chain that is not paralleled by other chains can occur only if substitutions happen in a loop. An example of a loop involving 3 orders is depicted in Figure 4.5. In this situation, order a successfully substitutes order c in the plan of the first truck, and then order c substitutes order f in the plan of the second truck. Subsequently, order f substitutes order a in the first truck, and order a substitutes order c in the second truck.

Given Step 3a in Algorithm 4.1, even if $\varepsilon = 0$, the first substitution can only happen if order a requires less empty travel in the same position in the plan of the first truck than order c. We say that the first truck prefers order a over order c, which we denote as $a \prec_1 c$. The second substitution can happen only if $c \prec_2 f$, the third only if $f \prec_1 a$, and the fourth only if $a \prec_2 c$.

Since the preference relation as we defined it is transitive, it follows that $f \prec_1 a \prec_1 c$, and that $a \prec_2 c \prec_2 f$. Now, the next substitution in the loop would be the substitution of order f by order c in the first truck. This, however, cannot happen, since f is preferred more by the first truck than c, and Step 3a of Algorithm 4.1 discards this alternative. Either an insertion is found

as the best alternative, or order c is rejected.

The crucial point in Algorithm 4.1 is Step 3a. It renders the substitution chain into a sequence of solutions of ever increasing values. Since the values of subsequent solutions must increase, loops cannot occur. This case is reinforced by the use of $\varepsilon > 0$.

Regarding the case when more than one substitution chain is active at a time (multiple orders arrive at the same time, and they start their initial auctions simultaneously), substitutions in a chain can be influenced by changes introduced in some other chain. Normally, job a cannot substitute job b, if b substituted a earlier. This is because of Step 3a in the algorithm. However, if another chain changes the neighbors of b, a can substitute it, because it might be a better solution in the new situation than b. What happens is that the number of possible substitutions is increased by the changes introduced by other chains. Nevertheless, since other chains are also finite, the increased number of substitution options is still finite. ■

The ε parameter has an important effect on the algorithm. Its value determines the length of the substitution chains. If $\varepsilon = 0$, substitutions may produce all $N!$ permutations of the N containers in the plans of trucks in a more than exponential number of steps. If ε is a very big number, it will disallow any substitutions, reducing the length of substitution chains to zero. In the following, we elaborate on the different settings of ε, and the corresponding bounds on the length of substitution chains.

Proposition 4.4 *There are minimal- and maximal-cost allocations of N orders to K trucks. The difference of these minimal and maximal costs is the maximum possible improvement that can be achieved by reordering the containers from the maximal cost order to the minimal cost order. Let us call the value of this maximum possible improvement* imp_{max}. *Then the length of the substitution chain* ($len_{sc}(N)$) *occurring during the* N^{th} *auction depends on* ε *as follows.*

$$len_{sc}(N) = \begin{cases} O(N^N) if\ \varepsilon = 0 \\ O(1) \quad if\ \varepsilon = \frac{imp_{max}}{m}, m \in \mathbb{N} \\ 0 \qquad if\ \varepsilon \geq imp_{max} \end{cases} \tag{4.8}$$

PROOF To calculate the value of $len_{sc}(N)$, it is useful to think about the the plans of the trucks as if they were chained together into one long route. When the Nth container arrives, this long route consists of $N-1$ containers. In the beginning, let us put the Nth container in a special location denoting rejection. When substitutions happen in a chain, in every round, one of the containers in the long route and the container at the rejection position is swapped. At the end of the substitution chain, the container at the rejection position can either find an insertion location extending the long route, or it is rejected.

In every round a new solution is generated with a different value. Let us call the set of solutions that can be reached from a given solution by swapping one order in the long route and the order at the rejection position a *substitution neighborhood*. The process represented by the substitution chains is a greedy search in the substitution neighborhood, where in every round the best swap is selected. This follows partly from the bidding mechanism, and partly from the auctioning. Truck agents submit the best bid found during the search for an insertion or substitution position within the trucks. Additionally, the auctioning order agent selects the best of the bids. Thus, the best of all possible swap moves is selected.

The role of the parameter ε is to cut the number of possible moves in every round. If $\varepsilon = 0$, all possibilities are considered, and the search considers all $N!$ solutions in $O(N^N)$ steps. If $\varepsilon \geq imp_{max}$, then there are no possible swaps that the search can perform in the first round, therefore the length of the substitution chain is zero. In the middle case, when ε is set to $\frac{imp_{max}}{m}$, such big jumps in solution value are required that the process cannot have more than m, that is $O(1)$ jumps. ■

The insertion-substitution algorithm enables truck agents to break contracts in favor of more beneficial contracts. This helps to overcome some of the inefficiencies of the insertion heuristic, and therefore provides more efficient plans. The following two sections describe two additional techniques that agents can use to further improve the solution. Both of these techniques are distributed versions of central heuristic algorithms that were adapted to our agent system.

Random Reallocation

Whenever the plan of a truck agent changes, (in principle) any order agent has a chance that the truck agent with the new plan can now transport it at a lower cost than their current truck agent. Rejected order agents receive notifications about such events, therefore they can try to close a deal with the truck agent whose state has changed. Since it is also useful for order agents that already master a contract to try to find better options, they use a randomized algorithm to search for those options.

Every order agent has a timer that goes off according to exponentially distributed random intervals with a mean value of μ_r. Whenever the timer of an order agent goes off and the agent has a contract, it tries to reallocate itself in the following way. First, it sends a message to the truck agent it has a contract with to prevent the truck from transporting it. The truck agent puts the order *on hold*, and reports the current insertion cost of the order to the order agent. The current insertion cost of the order may be different from its cost when it was inserted, because the previous and the following order in the truck agent's plan might have changed in the meanwhile. Then the order agent re-auctions itself to the other truck agents to see if any of them offers a better price. The order agent only accepts new offers that are better than its current insertion cost. If the order agent finds a cheaper option, it breaks its current contract and makes a new one with the new winning truck agent. If not, it sends a message to the current contracted truck agent in order to remove the hold allowing the truck to transport it. The steps of the algorithm are summarized in Algorithm 4.2.

Time and messaging complexity of the random reallocation algorithm equal the time and messaging complexity of the auctioning algorithm of order agents. The extra steps and messages required to put the order on hold (to stop the truck executing it, while the algorithm runs), and to report back the current insertion cost of the order add only a constant factor to both time and messaging complexities. Since an order agent runs Algorithm 4.2 periodically, according to a given exponential distribution, the number of runs is limited by a constant ($\frac{\text{planning horizon}}{\mu_r}$). This means that the complexity of Algorithm 4.2 run by an order agent periodically, is the complexity of the auctioning algorithm multiplied by a constant, which is therefore the same as the complexity of the auctioning algorithm.

By periodically attempting a reallocation, order agents check new possibilities that arose from truck-plan changes since the last attempt. In fact, order agents handle all new possibilities

Algorithm 4.2 Reallocation

1. The order agent sends a message to its contracted truck agent to prevent the truck from transporting it.
2. If the transportation of the order has not started yet, the truck agent puts the order on hold, and sends back its current insertion cost.
3. The order agent re-auctions itself among the other truck agents and collects offers that are better than its current insertion cost.
4. The order agent sorts the list of collected offers from best to worse, and iterates through the list.
   (a) Notify the next truck agent on the list that it won the auction.
   (b) If the truck agent accepts the new contract, the order agent notifies the previous truck agent that it leaves. The previous truck agent removes the order from its plan, and the algorithm stops.
5. If there are no offers left (or the list was empty because no new offer was better the current one) the order agent sends a message to the currently contracted truck to make it remove the on-hold status and transport it according to schedule.

in a periodic batch, in a decentralized manner via auctioning. Whenever they find a cheaper truck, they initiate a change not considering the problem that they might cause to their current truck. Similar to order agents, truck agents also actively try to improve the solution. The next section discusses a decentralized algorithm that exchanges orders between pairs of trucks.

Exchange of Orders

An efficient way to improve solution quality in vehicle routing problems is to try to exchange jobs between trucks. In the realm of local search heuristics, as explained in Section 2.2.1, a fairly large neighborhood is defined by the b-cyclic k-transfer move. Such a move shifts k jobs from each vehicle to the next one in a circular permutation of b vehicles. A full search of that neighborhood is only possible, with respect to computation time, for conservative k and b values. Our decentralized algorithm of order exchanges searches a neighborhood similar to a 2-cyclic k-exchange neighborhood in a randomized fashion. In the cases on which we tested our algorithms, the value of k never exceeded three.

Like order agents, truck agents also have a timer set to fire at exponentially random intervals with a mean value of μ_e. When a truck agent's timer goes off, it initiates an order-exchange procedure. It first checks if there are any orders in the plan that are not executed yet. Such orders are subject to exchange with another truck. The truck agent puts the first exchangeable order 'on hold', to prevent it from being executed. Then it copies the relevant part of the plan and sends it to another truck. In principle, truck agents can apply sophisticated heuristics to choose a partner truck. Geographical coordinates, personal preferences of trucks, or business considerations of the company can be taken into account. However, these heuristics rely on the availability of

appropriate information regarding all (or a subset of the) trucks. Maintaining such information implies the aggregation or centralization of data. As one of our goals was to design and test an agent system in which no information is concentrated, our truck agents do not discriminate in choosing a partner truck; selection of a partner truck is made randomly.

The truck that receives the chunk of plan extracts a similar sub-plan from its own plan. It also puts the first exchangeable order 'on hold', and then searches the two sub-plans for special k-exchanges that improve the solution regarding both sub-plans. The algorithm considers only those k-exchanges, where k consecutive orders are exchanged. First it tries to exchange single orders ($k = 1$), then chains of two, three, etc. orders. The maximum of k is provided by the length of the shorter sub-plan. This search returns the exchange combination of orders that yields the highest saving for the *combination* of the two trucks. If a solution is found, it is reported back to the initiator truck.

If the initiator receives a certain order-exchange combination from the responder truck, it implements the exchange in its plan, and sends new contracts to newly assigned order agents. The last step of the initiator is to send the expired contracts of the exchanged orders to the responder truck. To conclude the procedure, the responder also implements the order exchanges in its plan, and sends new contracts to the newly received order agents.

The order-exchange algorithm can be explained as a four-step negotiation between the initiator and the responder trucks. Algorithm 4.3 summarizes all the steps.

It is important to note that the above described order-exchange algorithm requires cooperative truck agents. First, truck agents are required to reveal parts of their plan to other truck agents. Second, the responder agent searches for an exchange that maximally decreases the costs of the two truck agents together. Such considerations prohibit the direct application of this algorithm in a competitive agent system.

The most computationally expensive part of Algorithm 4.3 is the search for the best k-exchange. During the search, first all the orders of one of the sub-plans are checked against single orders of the other sub-plan. Then pairs of consecutive orders, in one sub-plan, are checked against consecutive pairs of the other sub-plan. In every round, longer and longer exchanges are checked until, in the end, the full sub-plans are compared. The following proposition bounds the number of steps by a polynomial, and the number of messages sent during the computation by a constant.

Proposition 4.5 *Algorithm 4.3 finds the best k-exchange between two truck agents that improves their plans in $O(N^3)$ steps. The messaging complexity of the algorithm is $O(1)$.*

PROOF As for the first part of the proposition, time complexity of all but the second step of Algorithm 4.3 is constant. In the second step, a search for the best exchange of any length is performed.

The number of different exchange lengths (k-exchanges) tested equals the length of the shorter sub-plan, thus it is $O(N)$. Searching for the best exchange given a particular k consists of checking the exchange of each k length subsection of one of the sub-plans by each k length subsection of the other sub-plan, which can be done in $O(N^2)$ steps. The search in the second step of Algorithm 4.3 is done in $O(N^3)$ time, showing that the algorithm is polynomial.

The messaging complexity is less complicated, the algorithm requires passing only four messages, which is $O(1)$. ■

Algorithm 4.3 Order Exchange

The Initiator truck

1. puts the first exchangeable order 'on hold',
2. copies the segment of its plan that contains only exchangeable orders, and
3. sends this segment of length $planlength_1$ to a randomly chosen other truck.

The Responder truck

1. also puts its first exchangeable order 'on hold',
2. makes a $planlength_2$ long copy of the exchangeable part of its plan,
3. search for the best exchange where k consecutive orders are exchanged for $k = 1,2,\ldots \min(planlength_1, planlength_2)$, and
4. sends back the best exchange combination to the initiator, if any.

The Initiator truck

1. implements the proposed exchange,
2. sends new contracts to the newly acquired order agents, and
3. sends back the expired contracts of the exchanged order agents to the responder.

The Responder truck

1. implements the proposed exchanges in its plan, and
2. sends new contracts to the newly acquired order agents.

Table 4.1: Complexity of algorithms used by the agents.

Name	Time complexity	Messaging complexity
Initial auctions	$O(N^2 \log K)$ $\Omega(NK \log K)$	$O(N^2)$ $\Omega(NK)$
Insertion with substitution	$O(N^2)$	0
Substitution auction	$O(K\log(K)N^N)$ if $\varepsilon = 0$ $O(K \log K)$ if $\varepsilon = \frac{imp_{max}}{m}$ 0 if $\varepsilon \geq imp_{max}$	$O(KN^N)$ if $\varepsilon = 0$ $O(K)$ if $\varepsilon = \frac{imp_{max}}{m}$ 0 if $\varepsilon \geq imp_{max}$
Random reallocation	$O(N^2 \log K)$ $\Omega(NK \log K)$	$O(N^2)$ $\Omega(NK)$
Order exchange	$O(N^3)$	$O(1)$

Table 4.1 summarizes the time and messaging complexity of the algorithms that the agents use. All algorithms are polynomial in the number of orders (N) and the number of trucks (K) except the substitution auctions. As discussed at the substitution chains, these auctions can occur more than an exponential number of times, depending on the setting of the ε parameter. In the experiments, we simply constrained the costs (and therefore the bids and prices) to be whole numbers, and thereby set ε to be greater than 0. This means that the maximum number of substitutions equals imp_{max}.

The result of these algorithms is a set of plans that describe the routes each truck must travel. The next subsection explains the manner in which an on-line environment effects these heuristics.

4.1.2 Agent Operations in an On-Line Setting

The agent system consisting of truck and order agents, as described in the previous sections, can compute plans and is capable of changing those plans when new information is revealed. In our experiments, the agents send their plans to a simulator, where a corresponding virtual truck will execute the received plan. Execution of a plan in the on-line problem setting, therefore, means that the corresponding truck will simulate driving along the predefined route. Whenever the truck arrives to a service location as defined in its plan, it changes to execution mode (pick up, delivery, or return), spending the predefined amount of time at the location. When the execution time is over, the truck either starts traveling again towards its next destination, or if no next destination is defined, it stays at the current location and changes to an idle mode.

Operational state changes of the simulated trucks are monitored by the agents as a means to handle events caused by any type of uncertainty. New job arrivals are implicitly handled; order agents do not start their first auction before their designated arrival time. Other types of uncertainties are revealed to the agents by the status reports of the simulated entities. Whenever simulated trucks start, stop activities, or break down, they send a status report to the corresponding truck agent. When a truck agent experiences a deviation from its plan, it tries to adjust the plan to the new situation. Sometimes simply shifting the orders in time solves the problem. If any orders become infeasible due to time-window violations, then the truck agent removes the infeasible orders from its plan. Similarly, when a truck breaks down, all non-handled orders are released. The released orders experience a removal similar to a substitution, and they start a new auction to find another truck agent. In this way, all events are handled and the agents continue their improvement efforts as before.

We now turn our attention to the centralized approach that is studied in comparison to this agent system.

4.2 Centralized On-line Optimization

At the base of the on-line optimization method is a mixed integer program that is solved periodically to obtain a solution in each decision period. The MIP that models the “pickup delivery and return problem” as a “pickup and delivery problem” is defined in Section 3.3.2. In the following we describe how this model is applied iteratively to solve problems with uncertainty.

4.2.1 The On-line Approach

In order to provide a fair comparison with the agent-based approach, the MIP is manipulated for use in on-line operations. In our on-line approach, the MIP is invoked at fixed time intervals (30 seconds). At each interval, the full and current state of the world is captured, and then encoded in the MIP. This "snapshot" of the world includes information on all jobs that are available and in need of scheduling, as well as the forecasted next available location and time of all trucks (as derived from their current jobs). The MIP is then solved via a call to an external solver. The solution returned by the solver is parsed and any jobs, that are within two intervals (i.e. 60 seconds) of being late (i.e., missing the time specified by δ_i in the latest plan), if travel is not commenced in the next interval, are permanently assigned. Any jobs that were designated for rejection in the solution are permanently rejected only if they are within two intervals of violating a time window; otherwise they are considered available for scheduling in a subsequent interval.

If the solver cannot find an initial feasible solution in any interval except the first, then the plan from the last feasible interval is invoked and parsed to fix assignments and rejections as described. If the solver cannot find an initial feasible solution for three consecutive intervals (90 seconds), then one job is selected for rejection based on the following hierarchy:

1. a job that arrived since the last feasible plan was made.

2. a job with a loaded distance less than or equal to 13000 seconds.

3. a job that is 30 minutes away from the end of its time window.

4. a random job.

The first interval is handled differently, since in this case there is no previous feasible solution. In the first interval, the solver is allowed to run as long as an initial solution is found. If this happens within 30 seconds, then the solver is allowed to search further for a better solution until the 30 seconds elapses. Otherwise, this initial solution is parsed for assignments.

The iterative procedure continues solving problem instances and parsing solutions in this fashion until the end of the working day, at which point all jobs have been served or rejected. Notice that jobs may be rejected as a result of being rejected in subsequent MIP solutions or as a result of the solver's failure to find a feasible solution within the decision horizon.

Solving the MIP at every decision epoch is an NP-hard problem. The external solver employed in our solution needs, in the worst case, an exponential number of steps to find an optimal solution. Execution in an on-line manner, that is being invoked at fixed intervals, does not change this. The on-line optimization algorithm proposed in this section produces a solution to on-line problems in an exponential number of steps.

This algorithm implements the strategy of being as optimal as possible in every situation. According to Yang et al. (1999), this method is competitive with simple heuristics in terms of quality, although they do not study robustness. In the following sections, we point out the key differences this approach bears to our distributed agent-based method.

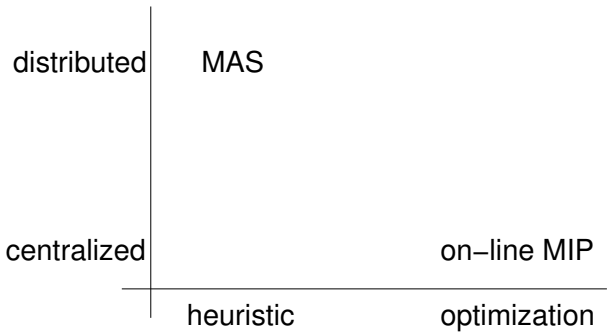


Figure 4.6: Key differences of the on-line optimization and the agent-based methods.

4.3 Analysis of the Two Approaches

Deterministic reactive planning approaches to vehicle routing problems with uncertainties can be categorized along two dimensions: (i) whether they try to make an optimal decision every time they need to react or not, and (ii) whether they make these decisions in a centralized or a distributed way. The agent-based solution discussed in Section 4.1 and the centralized solution in Section 4.2 are at the opposite ends of both scales (see Figure 4.6).

Our multi-agent system (as all multi-agent systems) solves the problem in a distributed way by solving several smaller problems (inserting, substituting, reallocating or exchanging orders) and combining the solutions (via auctions, and the exchange protocol). In comparison, the on-line MIP approach captures the actual state of the world in a central mixed-integer program, and solves this problem by a single sequential algorithm.

The centralized approach uses an optimizer that is allowed to run for only a fixed amount of time. Decisions made by it are optimal, whenever this time is enough to produce an optimal solution. Otherwise, if a feasible solution is found, it can be regarded as an approximation. In case not even a feasible solution can be found within the given time, this approach cannot make any new decisions. In comparison, the agent-based solution uses heuristic algorithms, therefore every solution derived is suboptimal. On the other hand, using fast algorithms, the agents suffer much less from the computational complexity of the problem. As we have shown in the previous sections, all agent algorithms are polynomial in the number of orders and vehicles, while the on-line approach requires an exponential number of steps.

The agent approach can be compared to a group of hunting dogs; not too efficient as individuals, but very flexible as a group. The central optimization is more like a tiger using brute force to reach its goals. When comparing the two approaches, we ask the question whether the flexible group, or the strong individual survives better. Our hypothesis is that there are certain circumstances that favor the strong individual, while others favor the flexible group. Through experimentation we uncover the critical level of uncertainties that cause one or the other approach to be more successful.

A further difference between the two approaches lies within their reaction mechanisms. The agents receive external triggers as reports on the execution arrive. They react on these triggers immediately by adjusting the plans, and reallocating orders via auctions if necessary. The on-line optimization approach, however, runs the optimizer periodically. Any change that happens in one period is considered only in the next period. Such a batch processing is efficient in the sense

that several events can be handled in one run, but inefficient in another sense that events are only handled in the next decision period.

By adapting and developing these approaches, our goal was to provide a distributed and a centralized transportation planning mechanism that can generate state-of-the-art solutions to on-line truckload vehicle routing problems. Although for our comparison study we use the selection of mechanisms described in this chapter, we acknowledge that there are other mechanisms that could be considered, or even that these mechanisms could be improved further. Therefore, a discussion of such future directions is provided in Section 7.3.2.

4.4 Summary

Our goal in this thesis is multifold. First, we want to introduce a general framework for considering uncertainties in vehicle routing problems. Second, we aim to provide a quantitative measure of robustness of logistical planning methods for such vehicle routing problems with uncertainties. Finally, our goal is also to present a comparative study of a centralized optimization and distributed agent method based on this robustness measure. To help the reader understand the details, we first provided an overview of vehicle routing problems, together with centralized and distributed solution approaches. Then in Chapter 3, we introduced our framework of uncertainties in vehicle routing problems, as well as our measure of robust transportation. In this chapter, as a contribution to Research Question 2, we described the planning methods used for the comparative study.

We started with the distributed agent system. We discussed the underlying agent model, and the algorithms agents use to make local decisions and to coordinate these decisions. Our contribution to the state of the art in this regard is an extension to the classical insertion heuristic, the so called insertion-substitution heuristic. This algorithm facilitates the reconsideration of earlier inserted orders in the trucks' plans, and thereby overcomes an inherent problem of the insertion heuristic: its sensitivity to the order of insertion. In addition to the insertion-substitution algorithm, we described the adaptation of two improvement heuristic algorithms to a distributed setting: the reallocation and exchange of orders.

After the agent-based solution, we described our adaptation of an on-line optimization method for truckload vehicle-routing problems with uncertainty. In comparing the two approaches, we concluded that fast algorithms of the distributed agent approach are likely to be more robust in cases heavily burdened with uncertainty. In relatively simple problems, however, with less dynamic events, the centralized optimization is likely to perform better.

In the following, we introduce an evaluation framework based on simulation that is capable of evaluating logistical performance under various types of uncertainty.

Chapter 5

Simulation Environment

5

In previous chapters, we discussed an extension to vehicle routing problems that enables the measurement of routing methodologies in the face of unforeseen events (see in Chapter 3). We called the property of being resistant to such events robustness, and extended the classical vehicle-routing problem definition with a framework of uncertain events. Then, in Chapter 4, we presented two solution methods, a centralized and a distributed one, that were designed to compete on vehicle routing problem instances extended with such events. In this chapter, we introduce our simulation testbed in detail, and describe the uncertainty scenarios that can be simulated to test planning methods. First, we discuss why we use simulation for comparing transportation planning methods, and define the desirable properties of the simulation environment. Then in Section 5.1, we describe the components of our simulation testbed, which is followed by a description of the kind of inputs the environment can handle in Section 5.5. The requirement analysis and an overview of the simulation framework were published by Máhr et al. (2011).

There are many different ways one can compare planning methods, and the choice of the evaluation framework can significantly influence the results. Using a simulation testbed represents a compromise between measuring different qualities of the plans and trying to execute them in a real environment measuring the results at the end of the day. The former, and more abstract method provides an easy comparison of planning methods. Such comparisons based on plans are frequently published in literature, for us the most relevant of these comparisons is in Mahmassani et al. (2000) and Yang et al. (2004). They compared on-line optimization to similarly operating centralized heuristics by measuring the plans at every decision epoch. In contrast, in our case, the competitors are agent-based heuristics, and we apply a different method to measure the performance. When making comparisons based on plans, it is difficult to define measurements that consider the effects of execution-time events. Such events, however, are implicitly considered by the method of actually executing the computed plans. Unfortunately, making real-life experiments with transportation planning algorithms is prohibitively costly. Simulations allow us to measure the dynamic performance of transportation-planning methods without the cost of navigating a real fleet of vehicles burning gasoline. The idea of comparing the spatial-temporal traces of vehicles to ascertain the quality of a routing policy is not new in the field of transportation (e.g. Sommer and Dressler, 2008). It is, however, more commonly used when considering personal vehicle routing as opposed to freight vehicle routing. There are also good examples of simulations, e.g. the Transportation And Production Agent-based Simulator (TAPAS) by Davidsson et al. (2008), where planning and execution are separated, although the evaluation of the results is not based on the simulation traces. In this chapter, we

describe a simulation environment that combines the properties of previous simulators to provide an improved environment for the comparison of dynamic transportation-planning algorithms.

To provide a fair comparison of transportation planning algorithms, any framework for comparison must accommodate certain requirements. We document the most essential of these requirements in the following three paragraphs.

The capability to consider unforeseen events. Our main goal is to evaluate transportation planning methods for problem instances with uncertainty. This means that in any instance, at the start of the planning horizon, there are events that must remain unknown to all planning methods until a specified point in time. For example, these dynamic or uncertain events may be defined in the instance data deterministically (e.g. at 3pm truck X breaks down completely), but the planning methods, should not know about these events before their occurrence (i.e. before 3pm). Thus, the simulation must be structured to ensure that, from the point of view of the planning method, the problem instances contain uncertainty.

The capability to accommodate different computation times. Different transportation planning methods have different run-time requirements. When solving dynamic transportation problems, the run-time of a planning method can severely influence the speed at which unforeseen changes can be dealt with. Algorithms with a long run time can potentially miss their chance to respond adequately to a given incident. To be able to make realistic conclusions about the behavior of planning systems in the real-world, it is essential to simulate the effect of long run times.

Therefore, we propose not using event-based simulations, because in such simulations there is usually no direct relation between computation time and simulated time, since simulated time is usually moved forward only after all computation related to an event has been completed.

The capability to separate planning and execution. To ensure a fair comparison of transportation planning algorithms, an evaluation method should use the same component to generate execution traces for each planning algorithm. This suggests that the planning and execution components should be separated (code-wise). This structure gives researchers flexibility by allowing reuse of the execution component (truck movement simulator) for each planning method. In turn, this lowers the development effort required to conduct the experiments and prevents drawing false conclusions from anomalies caused by different programming errors in the different execution codes.

To summarize, being able to include dynamic events during execution, explicitly consider the computation time needed by the planning methods, and reuse the same plan-execution component are, in our opinion, the essential properties of a fair evaluation method for transportation planning methods. In the next section, we introduce our test bed that meets all these requirements.

5.1 Simulation Test Bed

We start the introduction of our simulation test bed with a general overview of its composition. We present its architecture, and discuss how this simulation environment meets the requirements

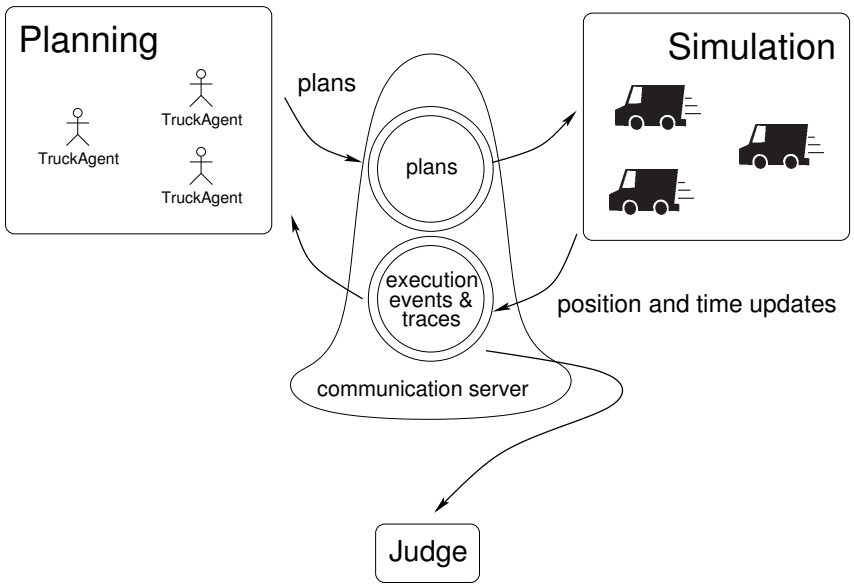


Figure 5.1: Evaluating a planner by measuring execution of simulated trucks.

stated in the previous section. In the subsequent sections, we discuss the components of the test bed in more detail.

The transportation problem with uncertainties defined in Section 3.2 requires a planning method to produce a minimal execution trace regarding a predefined measure (e.g. empty distance, total distance, etc). The execution trace is not the output of the planning method, but it is the result of trucks traveling on a road network marking the locations they visit. The traces of trucks are certainly influenced by the plans produced by the planning method, but also by the dynamic events happening during the execution of the plans. As a result, to evaluate the performance of planning methods in a routing problem with uncertainties, one has to simulate the traces trucks leave when following the produced plans, while the defined events take place. Figure 5.1 depicts our simulation test bed to make such measurements.

The architecture consists of a planning component, a simulation component, a communication server, and a judge. The separation of functions is (partly) dictated by the requirement of the *capability to separate planning and execution*. First, we discuss the planning component that implements the planning methods to be evaluated, and the simulation component that eventually produces the execution traces. In Figure 5.1, the planning component consists of agents cooperating in producing transportation plans. In addition to multi-agent planners, the evaluation framework can evaluate any kind of planner regardless of its internal structure. What is required from the planning component is that it computes plans, and sends them to the simulator through the communication server. The simulator executes those plans by simulating trucks visiting the locations in the plans. It reports the current state of the world back to the planning component (through the communication server), thereby facilitating the *consideration of unforeseen events*. It is important that the coordinates of the locations are interpreted the same way by the planner component and the simulator. In principle, the coordinates could be numbers relative to a planar coordinate system, with Euclidean distances between the points. Alternatively, coordinates can refer to geographical coordinates, and the distances can be given as distances in the road network. In our realization, we used the latter interpretation.

Aside from these spatial considerations, the planner and the simulation components should have the same sense of time too. This is important, because the simulator executes truck movements, and thereby generates traces in parallel to the planner component calculating new plans. As a consequence, the evaluation framework does not assume that either the computation of new plans or the communication of those plans are instantaneous, thereby fulfilling the requirement of the *capability to accommodate different computation times*. The longer the computation and communication of new plans take, the more time will be spent by the simulated trucks executing outdated plans. Notice that this architecture assumes that the simulator and the planner 'live' in the same time. This is trivial if both run in real time, but if one wants to simulate at a faster speed, the simulator and the planner should explicitly synchronize their times. To facilitate this, our simulator periodically reports its current time, the speedup factor, and its start-up time to the planner.

All communications between the planner component and the simulator are mediated by the communication server. The server maintains a set of topics that act as message channels. Other components can subscribe to topics, and they can send messages to topics. The messages sent to a topic are relayed to all components subscribed to the given topic.

Since the planner and the simulator communicate through the communication server, monitoring the messages is trivial by listening to the communication server. The evaluation system saves all messages in order to perform the measurements required to evaluate the given planning method. The sequence of position update messages from the simulator, for example, reveals the simulation traces and provides the ultimate source of information for evaluation. This saved execution log is processed by our last component, the judge.

The judge can perform various computations on the saved logs, which constitutes measuring the quality of the execution. For a given execution log and problem instance, the judge can compute the time (or distance) traveled empty by the trucks, time spent idle by the trucks, the aggregated time of lateness at deliveries, or any variety of user defined performance measure. To evaluate the routing method in the usual way, one should compute the measure corresponding to the objective function used by the planning method.

After this high-level overview of the components of the simulation testbed, the next sections describe the individual components in more detail. First, we discuss the design of the simulator, then two types of planning approaches: the agent-based planner, and the optimization-based planner.

5.2 The Simulator

The description of the simulator is structured in four parts. The first part introduces the global parameters that control the time and event handling in the simulator, and discusses its overall architecture. Then, Section 5.2.1 defines the messages the simulator can receive and interpret. Section 5.2.2 explains how the simulation of truck movements and the dynamic events are carried out, and Section 5.2.3 defines the messages that the simulator sends out during simulation.

The simulator component of the testbed runs as a standalone process started from the command line (or possibly from a script). The most important parameters it takes include the update period, the start date and time, and the time rate. The *update period* specifies how often

the simulator should update the state of the simulated vehicles and send this information back to the planner (through the communication server). The *start date and time* specify an artificial starting point for the simulation. The simulator maintains a virtual clock, which is set to this time point in the beginning of the simulation. All simulated entities query this virtual clock for the current time. The *time rate* specifies how many times faster or slower the virtual clock should tick as compared to the real clock. Values greater than one induce a speed-up in the virtual time, while values less than one mean that virtual time passes much slower than real time.

At start up, the simulator starts two threads: one to wait for incoming messages, and another to periodically update the state of the simulation. Both threads operate on a set of simulated vehicles. The incoming-message thread changes the states of the virtual vehicles according to commands received from the planner. The periodic-update thread changes the state of the vehicles as they move around according to the plans.

5.2.1 Incoming Messages

The incoming-message thread interprets four different kinds of messages. The *initialization message* contains a vehicle id (a number plate) and a location. This message makes the simulator create a simulated vehicle, and initialize it at the given location. The *terminate message* is the opposite, it makes the simulator remove a simulated vehicle given by its id in the message. When a *plan message* is received, the simulator looks up the corresponding vehicle, and modifies its plan. A plan is composed of a list of actions belonging to one or more orders. The actions are described by the type of activity (e.g. pick up, delivery, etc.), the location of the action, the time it has to be performed and its estimated duration. If the vehicle has no previous plan, it simply adopts the new plan. Otherwise, leaving the actions of the currently handled order in place, the rest of the plan is replaced by the new plan. Finally, the *on-hold message*, containing a vehicle id, is handled by flagging the corresponding vehicle as on hold. This is used when the planner is considering the modification of the plan of the vehicle, and it prevents the vehicle from engaging in the pick up of the next order. These four types of actions are performed asynchronously, when the corresponding messages arrive.

5.2.2 Simulation of Truck Movements and Events

In contrast to the asynchronous operation of the incoming-message thread, the periodic-update thread wakes up at predefined time intervals. In each turn, it iterates through the vehicles and updates their states. This includes recomputing their locations if the vehicles are moving, adjusting their activity if needed, and the actuation of incidents as prescribed in the configuration file. Locations are calculated using an external routing component that is based on the digitized road network of the Benelux states. This routing component provides address-to-coordinate translation, as well as shortest-path routing between locations. Actual vehicle locations are calculated using the shortest path, as provided by the routing component, and the time spent on traveling on the path.

Vehicles are always in one of three operational states. In the beginning, vehicles are in the *waiting state*. When they have a plan, and the time is such that they have to start moving towards the location of the next action in order to arrive there at the predefined time, they enter the

traveling state. When vehicles arrive at the location of their next action, they either enter the *execution state*, if the time is as prescribed in the plan, or the waiting state, if they arrive early. They stay in the execution state for the duration of the action they have to execute. After the action is finished, the trucks either travel on to the location of the next action, or change to the waiting state and stay, if there is nothing more to do or if the timing of the next action does not require them to start traveling immediately.

Incidents are unveiled according to their types. The two incident types simulated in our comparison study are the service-time variations and the truck breakdowns (release-time variations are simulated implicitly by order agents not being active before their release time). The former is defined as a change in the duration of a specific action (e.g. pick up) of an order. The latter is specified as a time when a given vehicle is removed from service. Both types of incidents are listed in a configuration file, and read in at start up. The prescribed service-time variations are looked up every time a new plan arrives. If the plan contains an action, for which a changed service time is defined, that action is executed with the changed service time. However, the planner is notified about such a deviation only at the actual finish of the action. The truck breakdowns are ordered in chronological order, and when the periodic updates reach or pass the time of the next breakdown, the given vehicle is stopped. In this case, the planner receives an immediate notification about the accident.

5.2.3 Outgoing Messages

To keep the planner and the judge up-to-date, the simulator sends out messages via the communication server. All messages are sent from the periodic-update thread. To allow the planner component to synchronize to the virtual simulation time, the simulator sends a *time message* at every periodic status update. This message contains the current time, the virtual starting time point of the simulation and the speed-up rate. When a vehicle is in the traveling state, at every location update a *gps message* is sent to inform the planner, and to allow the judge to measure the simulation/execution traces. The gps message consists of the id of the vehicle, the current (simulation) time, and the current location. Whenever the operational state of a vehicle changes a status-update message is sent. There are four different status-update messages: *waiting*, *traveling*, *action*, and *action-done*. The first three announce the waiting, traveling and execution state respectively, while the last one notifies the planner that the execution of the given action is finished. The waiting message contains the id of the vehicle, its location, and the time. The traveling message consists of the vehicle id, the time, the current location of the vehicle and its destination. The action message describes the action as in the plan, it contains: the vehicle id, the time the execution of the action started, its expected duration, its location, a reference to the order the action belongs to, and the type of the action. The action-done message contains the vehicle id, the reference to the order, the type of action that is finished and the time it was finished. Finally, when a vehicle breaks down as prescribed by an incident, a *breakdown message* is sent, containing the vehicle id, the location, and the time of the incident.

The simulator described in this section is capable of receiving plans, simulating vehicle movements according to these plans, and actuating incidents during the simulation of the movements. Status updates of the position and activity of the vehicles are sent periodically to allow the planners to monitor the state of the simulated world. How the different planners

are designed to make use of this information to adjust the plans is described in the following sections.

5.3 The Agent Planner

One of the two implementations of the planner component, in our comparison study, is an agent-based logistical planner. This planner has two separate layers: a general-purpose multi-agent-system layer at the bottom, and a transportation-agent layer above, which consists of a set of agents that implement the logistical model. The overall goal of the agent system is to solve transportation problems in an efficient and robust way. To this end, the role of the lower-level agent system is to provide an efficient base infrastructure for the agents, such as processors to execute algorithms, message delivery, or directory services to find each other. The agent implementations, building on these facilities, have the role of interacting to solve the given transportation problem: in a nutshell, they have to derive plans and send them to the simulator. In the following subsections, we describe the base multi-agent system as the foundation of the agent planner, and then the agents themselves, who concentrate on the communication processes that leads to an efficient solution of the transportation problem.

5.3.1 The Base Multi-Agent System

The base multi-agent system is implemented as a library, which allows any multi-agent application to easily make use of the services supplied. The library is called the *Communicating-Agent Library*[1]. In this section, we introduce the two main services of the Communicating-Agent Library, which are the communication infrastructure, and the thread-execution service (a controlled sharing of the processor resource). The communication infrastructure allows the agents to conduct multiple parallel overlapping conversations, which is crucial in multi-agent systems. The thread-execution service provides processor resources for the agents, so that they can execute their decision logic.

One of the goals when designing the agent library was to develop a multi-agent system that can be distributed over geographically dispersed hosts. Some of these hosts were intended to be on-board-computers of trucks, therefore the agent system had to allow embedded operation. This requires a small memory footprint and processor demand. To fulfill these requirements, the agent system was implemented in Objective C, which, being a minimal object-oriented extension of C, can handle such constraints. The following sections elaborate on the two key features of the agent library, and Section 5.3.2 explains how it is used to implement logistical agents.

The Communication Infrastructure

The communication infrastructure of the agent library is discussed in this section by first introducing the general concepts of multi-agent communication that the library implements, and the problems that arise. After that, the solution of the mentioned problems is explained, while introducing the components of the library.

[1] `http://chap.sourceforge.net/cal/doc`

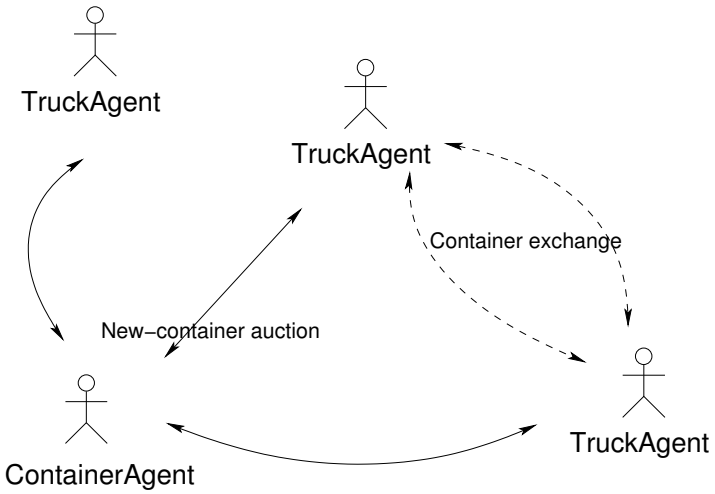


Figure 5.2: Agents in multiple communication processes.

As mentioned earlier, a main focus of the agent-system design is the communication of the agents. A general agent system is modeled by the library as a set of agents participating in multiple communication processes. One communication process has at least two agent participants that send and receive messages between each other. Examples are the auctions that are communication processes between a container agent and multiple truck agents, or the container-exchange procedure that is a communication process between two truck agents. An agent can participate in multiple communications, and send and receive messages within the context of the different communication processes. As depicted in Figure 5.2, a truck agent might participate in an auction and a container-exchange procedure at the same time.

A single communication process can manifest in several message exchanges between the agents. When an agent participates in multiple communication processes, deciding on which context a message belongs to can be a problem. The architecture of agents in the Communicating Agent Library is designed to assist the agent programmer in solving this problem.

Incoming messages from an agent are stored in the agent's own queue. Messages for new conversations, that is messages that are not replies to earlier messages, are queued in their order of arrival. When such a message is queued, it receives a sequence number that identifies the conversation for the agent. This becomes the receiver id of the message. When the agent replies to this message, the sender id of this message becomes the receiver id of the reply, and the receiver id of this message will be the sender id of the reply. This ensures that the replies are recognised by both parties as part of the conversation. (A description of ids and other fields of the `Message` class is provided in Figure 5.3.) Moreover, reply messages are inserted into the queue of the receiver agent according to an ascending order of the receiver-id. As a result, earlier conversations are prioritized over new conversations.

How an agent handles messages is implemented in `Context` classes. For different conversations, agents have different `Context` classes. For example, truck agents have different `Context` classes to handle new-container auctions and container-exchange conversations. Moreover, a single conversation may be handled by different `Context` classes at different agents. For example, a new-container auction is defined by two `Context` classes, one for the auctioneer, and one for the bidder. We call the method by which an agent finds out which `Context` it

field	type	description
sender	URL	the address of the sender agent in the form of 'agent-name@hostname'
receiver	URL	the address of the receiver agent in the form of 'agent-name@hostname'
name	string	an arbitrary name for the message
content	object	any object that contains the content of the message
sender-id	object	any object that identifies the conversation for the sender
receiver-id	object	any object that identifies the conversation for the receiver

Figure 5.3: The Message class.

should use to handle an incoming message *inter-context coordination*.

The main role of the agents' inter-context coordination procedure is to find out which `Context` to instantiate for the first message of a conversation. Once this is done, the `Context` object is stored to handle subsequent messages of the same conversation. For the first message, it is the responsibility of the agent programmer to implement a predefined abstract method of the agent to find the appropriate `Context` based on the actual state of the agent and the properties of the message at hand.

A `Context` object, once instantiated, is responsible for handling all messages that belong to the conversation. This is done by a simple form of *intra-context coordination*, by remembering the name of the `Context` method that should be called to handle the next message. In the beginning, the name of the method that handles the first message is remembered. When the first message has been handled, then the name of the second method is remembered, and so on. Every method that handles some parts of the conversation should conform to the following requirements. The methods should take a single `Message` object as an argument, they should set the `Context` (to remember the name of the next method), and they should return true or false, denoting whether the conversation continues or not (if false the `Context` object can be released).

To summarize, agent models in the Communicating Agent Library consist of `Agent` classes and `Context` classes. At a minimum, `Agent` subclasses should implement some kind of (inter-context coordination) decision logic that assigns `Context` classes to incoming messages of new conversations. `Context` subclasses should handle subsequent messages of a conversation by implementing methods of a special form that can accept the message, and specify what should be done with the next message.

The Thread-Execution Service

The other main goal of the agent library, besides providing a communication infrastructure for the agents, is to provide processor resources for the agents, and to ensure that different agents

can run in parallel, while different methods of the same agent cannot. To achieve this, the agent library uses a thread pool to execute `Context` methods. When there is an incoming message in the queue, the `Agent` object dispatches an execution job containing the first message and the corresponding `Context` object to the thread pool. When the `Context` object finishes handling the message and returns, it releases the thread back to the pool, making it available for other agents. If there are further messages to handle for the same agent, then a new execution job is dispatched just before releasing the thread. As a consequence, the jobs of different agents are multiplexed in the job queue of the thread pool. The number of threads an agent can use to handle messages is limited to one by always dispatching only one execution job with the first message. This guarantees that every message is handled sequentially, and that the `Agent` and `Context` methods do not need to synchronize their execution.

In this brief introduction of the Communicating Agent Library, we described the three main classes that are used to build a multi-agent model. These classes are the `Message`, the `Agent`, and the `Context` classes. We explained how the agents coordinate the handling of messages that belong to different conversations, as well as how the `Contexts` coordinate the handling of subsequent messages of the same conversation. The former we called *inter-contex coordination*, and the latter we called *intra-context coordination*. Finally, we described how the agent library handles the processor resource by the application of a thread pool. In the next section, we introduce the three agent types that implement our multi-agent model for the transportation problem we try to solve.

5.3.2 Transportation Agents

The purpose of the agents, built upon the foundations of the Communicating Agent Library, is to compute plans that are executed by the simulator, and to monitor the execution to be able to adjust (and resend) the plans as needed. Two agent types are discussed in the following section, the `Container` and the `ContainerTruck` agent types that implement the algorithms introduced in Section 4.1. A third agent type, the singleton `Communication` agent is also employed in the agent-based planner to facilitate the communication between the agents and the communication server.

There are three different conversation types, in which the `Container` agents and the `ContainerTruck` agents participate. These are the *new-container auctions*, the *container-exchange attempts*, and the *container-relocation attempts*. In addition, both agent types handle *execution-event* messages arriving from the simulator that are delivered by the `Communication` agent from the communication server. The following sections describe how the two transportation agent types handle messages in these four types.

New-Container Auctions

New-container auctions are conversations between a `Container` agent and multiple `ContainerTruck` agents. On the `Container` agent's side the context that handles an auction is called *`RFQ` (Request for Quotes)*. The first method of this context, called `findTruck` (see Figure 5.4), sends out RFQ messages to the `ContainerTruck` agents, and sets the second method, called `collect` to be called when answers arrive. This second method receives all

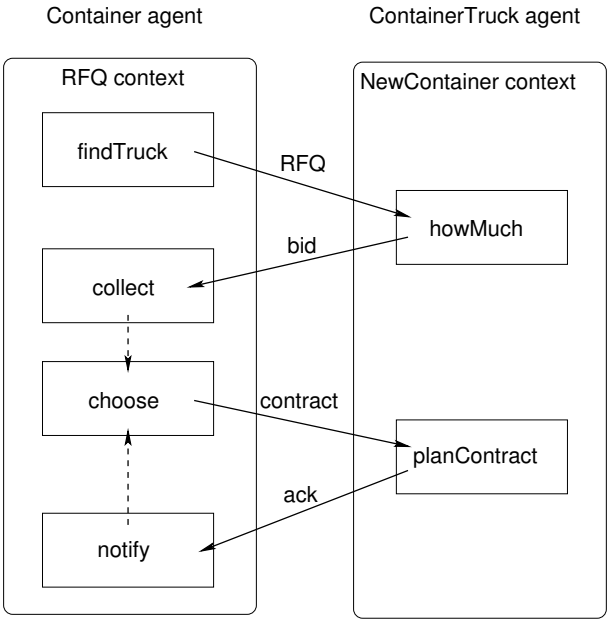


Figure 5.4: Agent contexts implementing the auction protocol.

the answers one-by-one and stores them. When all quotes are there, they are ordered, and the best quote is chosen as the winner (by the method `choose`). Then a message is sent to the winning `ContainerTruck` agent with a contract containing the second best price. The last method that is set to receive the answer from the winner is `notify`. If the answer is positive, this method notifies the losers, and stores the contract. If the winning `ContainerTruck` declines, the method reverts back to the previous method and tries to send a contract to the second best, and gets invoked again, when its answer arrives. This procedure continues until either a `ContainerTruck` agent sends a positive answer, or all `ContainerTruck` agents have been offered a contract. Either way, the `RFQ` context is released when the auction is finished.

On the `ContainerTruck` agent's side, the *`NewContainer`* context handles the messages of the auction. The first method of the context, called `howMuch` is invoked when the RFQ message of the `Container` agent arrives. This method performs a search for the cheapest insertion or substitution location in the plan of the agent concerning the `Container` whose RFQ was received. The cost of the best location is sent back as the quote, and a second method is set to handle the answer. If the answer is negative, meaning that the `ContainerTruck` did not win the auction, the auction is finished, and the `NewContainer` context is released. If the `ContainerTruck` agent wins the auction, then the second method of the `NewContainer` context, called `planContract`, is invoked. It checks if the `Container` can still be transported for the quoted price, since winning another auction in the meantime might have changed the plan in such a way that the quote is no longer valid. If the quote is still valid, the second method sends a positive reply. In the end, the `NewContainer` context is released.

Container Relocation

Even when a `Container` agent already has a contract with a `ContainerTruck` agent, from time to time it tries to find a better (cheaper) `ContainerTruck`. This is handled by the *`Relocation`* context of the `Container` agent. The `Relocation` context is a subclass of

the RFQ context, since looking for a cheaper ContainerTruck agent constitutes starting a new auction, leaving the currently contracted ContainerTruck agent out. In effect, the first method of the Relocation context sends a notification message to the currently contracted ContainerTruck agent about the relocation attempt. When the acknowledgement arrives, the second method starts the auction the same way as in the RFQ context, but leaves out the currently contracted ContainerTruck. The winner-selection method is overriden to consider only quotes that are better than the current contract, but otherwise the auction is handled the same way as in the RFQ context. When the auction is finished, the ContainerTruck agent is notified about the result.

At the ContainerTruck agent's side, the relocation attempt is handled by the *ContainerLeaves* context. This context is instantiated only at that ContainerTruck agent that initially has the contract with the Container agent. The first method of this context puts a lock on the Container that is under reallocation, in order to prevent the substitution of the Container (in favor of another Container) while it is engaged in searching for a better ContainerTruck. An acknowledgment is sent back to the Container, and a second method is set to receive the result of the relocation attempt. When the notification from the Container agent arrives, the second method either removes the Container from the ContainerTruck agent's plan, if the relocation attempt was successful, or simply removes the lock, and keeps the Container in the plan.

Container Exchange

ContainerTruck agents also try to further optimize their plans. During a container-exchange attempt, two ContainerTruck agents search for a locally optimal exchange of their containers. This is handled by *ContainerExchange* contexts. Interestingly, the same context (although different instances of it, of course) is used on both sides of the conversation, because both sides are ContainerTruck agents. One of the sides takes the *initiator* role, while the other takes the *responder* role. The initiator starts by locking those Containers in the plan that have not yet been transported, and sending that part of its plan to another randomly selected ContainerTruck agent. When the other agent receives this messages, it takes the role of the responder, and searches for a locally optimal exchange of Containers as described in Section 4.1.1. If an exchange is found, the list of Containers is sent back to the initiator, and both ContainerExchange context modify the plans of their ContainerTruck agents. After performing the exchange, both contexts send out new contracts to the exchanged Container agents. At the Container agents' side these messages are handled by the *ContainerExchanged* context, that stores the new contract.

Execution Events

Container and ContainerTruck agents must also handle messages that arrive from the outside world (the simulator). Simulated trucks send report messages to the execution-events topic of the communication server whenever they pick up, deliver, or return a container. These messages are relayed by the Communication agent that subscribes to this topic to the corresponding ContainerTruck agents. Each ContainerTruck agent has a single *ExecutionEvent* context that receives these messages. This allows the ContainerTruck agents

to monitor the execution, and signal when it deviates from the plans. The context checks whether the plan of the `ContainerTruck` is still feasible, and if not, initiates decommitment procedures. In addition, notifications about the activities are sent to the corresponding `Container` agents. At the `Container`-agent side, a similar context is used to handle these notifications, and store the actual state of the `Container`.

5.4 The Optimal Planner

To present how different planning methods can be evaluated by the framework, in this section, we discuss certain implementation details of the on-line optimization approach introduced in Section 4.2. The goal of this method is the same as the goal of the agent-based planner: create and adapt plans that result in minimal execution traces under uncertainty. To achieve this goal, the on-line optimization planner has to send plans out to the simulator through the communication server, and it has to be able to receive status updates via the communication server. Therefore, this planner has two independent threads of execution. One is computing the optimal plans periodically (*computational thread*), and the other one receives the status updates (*communication thread*).

Both threads operate on a set of data structures that describe the state of the world. These include the orders, the trucks, and the plans of the trucks. The order data are set once from the input in the beginning, and are hardly changed during the simulation. Only the state of an order may be changed to *open*, if it is available for planning, or *assigned*, if it is planned and under execution. The truck structures are initialized from the input, and the trucks' states (next available time and location) are maintained throughout the simulation. The plans are calculated periodically, and are fixed only at the latest time execution must begin to avoid lateness (see Section 4.2).

The plans are calculated by the computational thread based on the data describing the current state of the world. The data on orders and trucks are substituted in the mixed-integer model, which is then solved by an optimizer, resulting in an assignment of orders to trucks. The assignments that have to be executed immediately to finish in time are fixed as plans and are communicated to the simulator through the communication server.

Parallel to the computational thread, the communication thread listens for incoming messages from the simulator. Time messages are handled by setting the clock according to the time in the message. Whenever a status update for a truck arrives, this thread checks whether the execution is going according to the plans. If a truck is delayed as compared to the plan, the plan is adjusted in such a way that every action is shifted by the amount of the delay. When a truck-breakdown message arrives, the corresponding truck is marked as unavailable (its next available time is set to infinity), and all non-executed orders are marked as open, and will therefore be considered in the next periodic optimization round.

Operating in this manner, the communication thread ensures that the data structures always contain the actual state of the (simulated) world, and therefore the computational thread can compute the most relevant plans at any time. This mode of operation integrates the on-line optimization planner into the proposed evaluation framework for transportation problems with uncertainties. Furthermore, this integration is independent of the actual MIP model, or MIP

solver used. In our case, the MIP is the one defined in Section 3.3.2, and it is solved by the SCIP[2] solver. Nevertheless, a wide range of transportation models solved by any solver can be evaluated this way.

After introducing the functional components of the evaluation framework, the next section describes the possible inputs that the framework can interpret.

5.5 The Simulation Input

There are two different types of inputs that the components of the evaluation framework can/should interpret. The first type describes the static structures of the transportation problem, namely the trucks and orders. The second type defines the dynamic nature of the problem, in the shape of the events that cause uncertainty during execution. The following sections describe these two types of inputs.

5.5.1 Trucks and Orders

Both the simulator and the discussed planning methods handle truckload (e.g. container) pickup and delivery problems. That is, trucks are assigned only one order for any given time. This is also what the simulator expects. Trucks in the input have to be defined simply by an identifier (e.g. number plate) and their initial location. This input is used both in the planners and the simulator to initialize the simulation.

The input of orders is more complex. In addition to an identifier orders also contain a description of the three stops required to complete any given order. All stops are described by a location, and a time window. The location specifies where the truck should go, and the time window constrains the possible arrival time of the truck. The three stops represent the pick up of the order (container), its delivery to the customer, and the return of the container to a depot. The pick up and the return locations are not required to be the same. The order input is only relevant for the planners, the simulator does not directly handle the orders, only through the plans of the trucks.

Any location in the input should be given either as a postal address, or as a pair of latitude-longitude coordinates. Both the simulator and the planners use a navigation library that calculates routes between geographical positions on the road network of the Benelux states. Therefore the only requirement for a location is that the navigation library should recognise it as a point of the BeNeLux road network.

Time points in the input should be specified as absolute time points in the form of 'YYYY-MM-DD hh:mm:ss'. The planners synchronize their time to the simulator, and the simulator can be parametrized to start at any time. Therefore time windows, for example, can be set to start and finish on any given (possibly different) date(s), if the simulator is initialized to start its virtual clock before any of the time points in the input.

The trucks and orders, and space and time specifiers describe the static structure of a transportation problem. What makes a problem instance dynamic are the events that arise

[2] http://scip.zib.de/

during the simulation. The remainder of this section describes the different event types that the evaluation framework can possibly handle.

5.5.2 Simulated Events

Events that should happen during the execution of plans are defined as extensions to the input as described above. Some event types modify the values given in the static input, others simply add new parameters. In the following, we describe how the event types defined in Section 3.2 are simulated in the proposed evaluation framework.

Release-Time Uncertainty

To extend a static problem instance to a release-time uncertainty instance, one has to define release times for every order in the instance. When no release time is given for an order, it is assumed to be available for pick up at the start of the simulation. When a release time is given, the order is not considered by the planning methods until that time. That is, no plans computed before the given time will contain that order.

Service-Time Uncertainty

When a truck handles an order, it visits the three prescribed locations at times computed by the planning methods. Each visit takes a certain amount of time, for which there are default values built into the simulator. These default time intervals can be overwritten by specifying a *duration* parameter for each location in the input. The duration parameters are ignored by the planning methods (that consider the default values), but actuated by the simulator. This way, the variations of service times are unexpected to the planners who can consider them only after they happened.

Truck-Breakdown Uncertainty

The simulator can simulate accidents that render trucks unusable for a while, or for the rest of the simulation time. In the current implementation, trucks beak down totally, and they do not participate further in the transportation of the containers. To simulate such events, one has to extend the input description of the trucks by adding a *breakdown-time* parameter to trucks that should fail during execution. The value of the parameter should be an exact time. These parameters are ignored by the planners, who are informed of truck breakdowns by the simulator through a message. The simulator, however, registers those trucks that should break down, and stops simulating their movements at the defined time.

Travel-Time Uncertainty

In the simulator, virtual trucks travel along the path prescribed by their plans. The route that is followed is calculated by the navigation library used both in the simulator and the planners. The virtual trucks travel along the calculated routes at the maximum allowed speed for the given road. This information is included in the navigation library. It is possible, however, to change this value, and thereby simulate time-dependent travel times on selected road segments. To specify such variable travel times, one has to specify the possible travel speeds on any given road in

different time ranges (e.g. 6am-10am, 10am-3pm, 3pm-6pm, etc.) This can be done, for example, from data that is available for certain locations on the map. Alternatively, one could specify a single time-table for all roads.

Extending the input of the simulator with such road-speed definitions allows it to consider this information while simulating the truck movements. The planners, however, oblivious of these 'actual' travel speeds continue to use the maximum allowed speeds encoded in the navigation library.

Communication Uncertainty

When the planning methods calculate new plans, they send those plans to the simulator. The simulator then starts moving the corresponding virtual trucks along their path immediately. Presently, status and location update messages are sent from the simulator to the planners to keep them up-to-date. Since all messages pass through the communication server, it can introduce a consistent delay in the delivery of messages. To simulate communication difficulties this way, one has to be able to specify the time at which the delay of the messages should commence, and the topic of these messages. The communication server can then apply a message filter that matches the messages that should be delayed. With multiple such definitions it is possible to model different sources of communication delays (e.g. plan messages are delayed but status messages not, or the other way around).

5.6 Summary

In this chapter, we proposed a simulation environment that was designed to evaluate transportation-planning methods under uncertainty. To answer Research Question 1b, which is about the requirements of a proper comparison framework for transportation-planning methods under uncertainty, we collected the most essential requirements such a simulation environment should meet. These requirements are elaborated in the chapter, and a solution is proposed including implementation details.

One of the requirements relates to the simulation of truck movements. Since we believe that it is not possible to properly compare planning methods looking only at the plans they produce, the proposed environment includes a simulator component that produces execution traces in response to the plans generated by the planners. To evaluate different planning methods, these execution traces are used as a base of comparison.

Another requirement of the agent-based planner, evaluated in this environment, is that it should be possible to run it in a distributed manner, on the computers of competing business entities. In the transportation domain, agents represent companies that would not share internal planning information with each other on a central computer. Therefore, the Communicating Agent Library allows the agents to send messages to each other over the Internet. Furthermore, some of the agents were intended to operate in on-board-computers of trucks, in a very limited environment. That was one of the early reasons as to why a graphical representation of the agents is not included in the agent library, but forms a separate component. This need for distributed architecture drove the design of the test bed.

Having a distributed test bed, as in the requirement of the *capability to separate planning and execution*, has some important consequences. First of all, it is possible to replace different components of the simulation environment allowing for the comparison of very different approaches. For example, the planning component of the environment can be implemented by an agent-based heuristic or an on-line optimization planner, as in our case, but it is also possible to involve human planners in the control of the simulated trucks. Using this framework, one can even compare a dispatching method to a planning method. It is also possible to change the problem that the methods should solve. The same environment could be applied in other kinds of transportation problems. Moreover, replacing the simulator by real trucks would turn the test bed into a real transportation application.

Further consequences of the distributed environment are the possibility of a seamless integration of a monitoring component, and the possibility of experimenting with communication failures. Both of these advantages rely on all messages being sent through a communication server. By intercepting all messages at the communication server, it is possible to provide a graphical user interface, for example, to monitor the state of the (simulated) world. The communication server, being centrally positioned between the planning, monitoring, and simulating components, can introduce delays in message delivery with arbitrary patterns.

In addition to the aforementioned properties of the simulation environment, it has the *capability to consider unforeseen events* by allowing the simulation of different uncertainties, and the measurement of the performance of planning methods under these uncertainties. We discussed 5 different sources of uncertainty that can be simulated in the proposed environment. With the 5 uncertainty types, the simulation environment provides a complete evaluation environment for dynamic transportation planning methods. Finally, by simulating in (quasi) real time, the simulation environment fulfills the requirement of the *capability to accommodate different computation times*, since any planning methods have the same amount of time to react to events.

After introducing our simulation environment, and describing the uncertainty types that can be simulated in the test bed, we turn now to the set up of the experiments and the results that were produced in the various settings.

Chapter 6

Experiments and Results

6

In this chapter, our goal is to reveal how the distributed agent-based and centralized optimization-based planning methods perform on transportation problems under uncertainty, as defined in Chapter 3. We wish to find out which method, in the presence of uncertainty, is more cost effective, and which one is more robust, as per the definition of robustness in Section 3.4. Additionally, we are interested in how the different uncertainty types influence the results of the planning methods. A method might be more sensitive to some uncertainty types, while more resilient against others.

Some of the results discussed in this chapter, specifically the cost-measure comparison of the agent-based and on-line approaches in release-time and service-time uncertainty instances was published in Máhr et al. (2010). This work was extended by the addition of truck-breakdown uncertainty instances, and the analysis of the results using our robustness measure in Máhr et al. (2011). Finally, the problem instances were contributed to the MPLIB collection and published in an ERIM report by Srour et al. (2010).

Using the simulation framework introduced in Chapter 5, we embed different combinations of the agent heuristics plus the on-line optimization method into the simulation environment, run experiments with a selection of the possible uncertainty scenarios defined in Section 3.2 (the exact definition of the tested scenarios are given in Section 6.1), and measure the execution traces of the simulated trucks to compute the overall cost and the robustness of the methods. All methods are given the same problem instances, and they all use the same truck simulator to ensure a fair comparison. Furthermore, the size of the problem instances are selected such that it is computationally feasible for all methods to handle the instances. This is important because the on-line optimization has exponential time and space requirements, and therefore it is sensitive to the input size. Our scalability experiments, as documented in Appendix A, shows that both method can handle instances up to 90 orders, above which the on-line optimization runs out of memory. Therefore, in the experiments presented in this chapter, we used instances that contained less than 90 orders.

The experiments are performed in two phases. First, all sensible combinations of the agent heuristics are compared on a restricted set of scenarios, then two selected combinations and the on-line optimization are tested on all defined scenarios. For each case, we present two box plots showing how the cost and robustness results change as the level of uncertainty increases. After the analysis of the performance of the methods in the uncertainty scenarios, we discuss the effects of the uncertainties on the methods. This is presented in the form of interaction plots that depict the individual, as well as the combined effects of the uncertainties on the methods. Such

plots are presented and analysed for the on-line optimization and the agent methods.

Given the nature of the two approaches that we wish to compare, we have certain expectations regarding the results of the comparison. We express these expectations in the form of hypotheses in the following paragraphs.

In the first phase of experiments, the agent heuristics are compared on a restricted set of scenarios. The insertion, substitution, relocation, and exchange procedures are combined in eight different ways to compete with the on-line optimization and the *a posteriori* optimal solutions in the static and the most uncertain release-time scenario. The goal of this comparison is to evaluate the substitution algorithm, that is to see how those algorithm-combinations that contain the substitution method perform. Since the substitution algorithm was designed to overcome one of the shortcomings of the insertion method, its dependence on the insertion order of the jobs, we expect it to improve the cost-based performance of the planning in the static scenario. At the same time, we expect it to perform similarly to the other agent methods in the dynamic scenario, where the insertion order of the jobs is fixed by the arrival times, and it is not possible to change previous contracts that are under execution at the time a new job arrives.

Hypothesis 6.1 (Substitution heuristics) *The substitution heuristic introduced in Section 4.1.1 improves the cost effectiveness of the agent-based method in the static scenario, while it does not have a negative effect in the dynamic scenario.*

In the second phase of experiments, two selected agent heuristics are compared to the on-line optimization on a wide range of scenarios including static problem instances. We expect the on-line optimization to be superior in the static cases, in which it can take advantage of the available information, and it can find near optimal solutions. However, fully optimized immediate solutions might make a bad use of the resources on the long run, and therefore we expect the on-line optimization not (or not that much) to be superior in the uncertain cases. There might be levels of uncertainties where the cost effectiveness of the agent heuristics equals or even surpasses that of the on-line optimization.

Hypothesis 6.2 (Cost results) *The on-line optimization outperforms the agent heuristics in static instances, in terms of costs, but its advantage decreases under uncertainty.*

Given our previous expectation on the cost effectiveness results and the property of the robustness measure discussed in Proposition 3.1, we expect the agent methods to be more robust than the on-line optimization method.

Hypothesis 6.3 (Robustness results) *The agent-heuristics are more robust than the on-line optimization according to Definition 3.9 of robustness.*

The question about the most influential uncertainty type is somewhat misleading, suggesting that different levels of different uncertainty types are comparable. Naturally, it is not so. One cannot compare uncertainty levels in general; it is difficult to tell whether an instance having 50% dynamic jobs contain more uncertainty than one with 5 truck breakdowns. Given a set of uncertainty levels of different kinds, it is possible, however, to tell which levels of uncertainty had the same effect *on a given method*. In the next sections we define different levels of release-time, service-time, and truck breakdown uncertainties. Given the fact that uncertain service times directly influence the lateness of the deliveries, and thereby the cost results, we expect that the highest level of service-time uncertainty has the strongest effects on the compared methods.

Hypothesis 6.4 (The strongest effect) *Lower levels of service-time uncertainty influence the results of any planning method in a manner similar to higher levels of release-time or truck breakdown uncertainties.*

The hypotheses listed above express our expectations about the answers of the questions we raised in the beginning of this chapter. In the rest of the chapter, we first describe our problem instances exactly, and then present the results of two simulation phases. We start, in Section 6.1, by explaining how the instances were generated from the data, and provide the exact parameter values that were used. Then, in Section 6.2, we discuss the results of the different agent heuristics, and select two of them to compete against the on-line optimization. In the main part of the chapter, in Section 6.3, we present the results of the pure- and mixed-uncertainty scenarios along with a discussion of the individual and mixed effects of the uncertainties on the results. Finally, the chapter concludes with a summary of the main findings.

6.1 Experimental Design

To generate problem instances, we used operational data from a mid-sized Dutch logistics service provider (LSP) engaged in the over-the-road transport of sea containers. For the LSP, the process of executing a container order starts with receiving an order, generally one day before execution is required. While the orders are often called in one day early, the company does not generally use this information to plan routes or establish schedules. This is due to the unreliable nature of the order information and the resulting uncertainty encountered during execution. An order is a customer request to the LSP for the pickup and transport of a specific container from a container terminal (in the case of an import container) to the customer, with delivery required within a certain time window. Arriving at the customer's requested location, the container is unloaded, and the empty container is brought back to a container terminal or empty depot. Planners at the LSP estimate one hour for pick up, and for delivery of a container, and half an hour for return. After returning the container, the truck is ready for its next order. The process is reversed for export containers. What adds uncertainty to this process is that not all containers are available at the time indicated in the order: either they have not physically left the ship at the expected time or they are delayed for administrative reasons, e.g. an unsettled payment or customs clearance. The LSP can only transport containers that have been released, and are allowed to leave the container terminal. For this reason it is hard to optimize the system in a traditional sense, since not all information is known beforehand, and will only become available at some point in time during the day.

The uncertainty in the release times of containers makes this case suitable for dynamic planning methods. The release of a container can be modeled as a new order arrival in classical dynamic transportation problems. Such arrival events can be complemented with other dynamic events to form the input of the simulator. Thus, our test instances are partially based on real data, since the orders come from the company's database, but they are also partially generated, since the dynamic events are generated according to certain random distributions.

In all, we were given the execution data from January 2002 to October 2005 as well as the data from January 2006 through March 2006. We could not, however, use this data in its raw form. We first had to make multiple corrections to the customer address fields, as many addresses

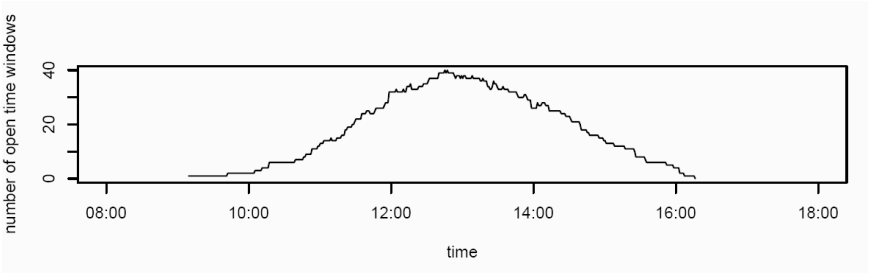


Figure 6.1: Number of open time windows throughout the working day.

referred to postal boxes and not to the physical terminal locations. After cleaning the address fields, we then extracted a random sample of jobs from the original data-set in order to generate a set of 33 days with 65 jobs and 40 trucks per day.

To standardize the data for our experimental purposes, we specified time windows at all locations as follows: terminals are open for pick up between 6AM and 6PM, and for return between 6AM to 5:59AM on the next day. The wide return time windows reflect the practice that trucks can bring containers to the terminals on the following day, if they were too late on the same day. Delivery time windows are set to two hour intervals, and their start times are distributed uniformly over the working day, between 8am and 5pm. Figure 6.1 displays the number of open time windows at any time point of the working day. Since time windows open regularly and stay open for two hours, the number of open time windows gradually builds up and reaches the maximum between 12AM and 2PM. After that, the number of open time windows, and therefore the number of urgent jobs, decrease. Given the variation in customer locations, the workload per day varies similarly. On average each job requires approximately 4.2 hours of loaded distance. When the routing is optimal the average empty time per job is approximately 25 minutes.

Figure 6.2 summarizes the fixed and free parameters of the simulations. These parameters belong to both the vehicle routing problem that the different methods try to solve and the solution methods themselves. When a parameter value was fixed throughout the simulations, it is explicitly mentioned in the figure. Those that were varied are typeset in italics. Most of the vehicle routing parameters that were fixed relate to the real-world problem and the data-set that we used to generate the instances. The number of trucks and orders, the service times, the delivery time windows, or the start time of the daily operations were set according to what the human planners revealed about the case. The simulation speed was chosen to balance between the requirements of providing enough time for the planning methods to react to events, and of keeping the total simulation time short. The particular choices of the parameter values for the solution approaches are more arbitrary. The reallocation and exchange frequency of the agent method (how often the agents try to improve the plans) was calculated to avoid processor contention among the agents by considering the number of agents and the speedup factor of the simulation. The decision horizon of the central optimization was set following similar considerations. According to some exploratory tests, 30 seconds seemed to be enough for the optimization to find a solution in most cases. Finally, the value of M for the optimization method must be larger that the longest distance in the problem, which is satisfied by the value 9000.

Given our interest in comparing the distributed agent solution and the centralized optimization

1. Vehicle routing parameters
   (a) number of trucks and orders are fixed to 40 and 65, respectively
   (b) delivery time-window length is fixed to 2 hours
   (c) for all containers, pick-up time is 1 hour, delivery time is 1 hour, return time is $1/2$ hour
   (d) delivery time window starts are uniformly distributed between 8AM and 5PM
   (e) start time is fixed to 6AM
   (f) simulation speed is fixed to $6\times$ real time
   (g) *parameters of uncertainty scenarios*
      i. *percentage of dynamic order release times*
      ii. *variation of service times*
      iii. *number of truck breakdowns*
2. Solution approaches parameters
   (a) Agents
      i. reallocation frequency is fixed to 1 try per 1 hour simulated time
      ii. exchange frequency is fixed to 1 try per 1 hour simulated time
   (b) Central optimization
      i. M value is fixed to 9000
      ii. decision horizon is fixed to 30 seconds real time, except the first decisions that are unlimited

Figure 6.2: Parameters of the experiments.

method on this pick-up delivery and return problem with time windows and different uncertainties, we further rendered our 33 days of data into scenarios capturing different types of uncertainties. In Section 5.5.2, a list of possible event types are discussed that can be handled by the simulator. In our experiments, we defined uncertainty scenarios using only a subset of those possible event types, namely the new order releases, service-time variations, and truck breakdowns. For each type of uncertainty, we defined a set of scenarios that differ only in the value of the parameter that describes the uncertainty type; e.g. service-time uncertainty scenarios differ only in the actual service times of orders. For each scenario, the 33 instances were adjusted to represent the level of uncertainty corresponding to the scenario. That is, for each uncertainty type, we defined a set of scenarios, each of which consists of 33 instances generated from the same 33 days of data. As a result, the simulation result for a scenario is an average of the results across the 33 problem instances.

6.1.1 Release-Time Uncertainty

As mentioned above, we model uncertainty in the release times of containers at sea terminals as uncertainty in the release times of jobs. Accordingly, we defined a job-arrival time for each container in every problem instance. The earliest job arrival was at 6AM; containers with such an arrival time we call *static jobs*. We call them static because they are actually known to the planning systems when they start planning for the day. We call containers with an arrival time later than 6AM *dynamic jobs*, referring to the fact that the planning systems need to incorporate these jobs into the plans during execution. When a container is randomly selected to be dynamic in a certain problem instance, we set its arrival time to exactly two hours before the start of its customer time window (i.e., four hours before the end of the customer-location time window, leaving slightly less than two hours on average before the latest departure time from the pickup location).

We generated five different scenarios with varying levels of release-time uncertainty. The varying levels of uncertainty were expressed in the percentage of dynamic jobs present in the instances. In the zero percent scenario (R_0), all jobs are known at the start of the working day, 6AM. In the 25% scenario (R_{25}), one quarter of the jobs (selected randomly from the 65 jobs) arrive two hours before their customer time window, and the rest are static. In the 50% scenario (R_{50}), half of the jobs are static and half of them are dynamic. In the 75% scenario (R_{75}), one quarter of the jobs are static. Finally, in the 100% cases (R_{100}) all jobs arrive in a dynamic fashion. These five scenarios cover the spectrum of dynamic job arrivals from all static jobs to all dynamic.

In scenarios of other (pure) uncertainty types, all jobs arrive at 6AM, therefore they correspond to an R_0 scenario with respect to job arrivals. For convenience reasons, we do not include the R_0 notation in the name of the other scenarios. Therefore any scenario without any explicit R_* notation implicitly refers to R_0.

6.1.2 Independent Service-Time Uncertainty

In scenarios with service-time uncertainty we define different service times for each service action (pick up, delivery, or return). While the planning methods consider the standard 1 hour

Table 6.1: Summary of bounds on randomly generated service times for independent service-time uncertainty scenarios, and the empirical mean M and standard deviation S of the generated datasets.

	Pick-up and Delivery Service Times			**Return Service Times**		
	Bounds [min]	M [min]	S [min]	Bounds [min]	M [min]	S [min]
S_{600}	[50, 70]	60	6	[20, 40]	30	6
S_{1200}	[40, 80]	60	12	[15, 50]	32	10
S_{1800}	[30, 90]	60	17	[15, 60]	38	13
S_{3600}	[30, 120]	74	26	[15, 90]	53	21

pick-up and delivery times, and the $1/2$ hour return times, the randomized service times are realized during execution.

Two different methods were used to generate service times: *independent* and *dependent*. In scenarios with independent service times, we generated the durations for all actions as random variables drawn independently from a uniform distribution. The boundaries of the uniform distributions for each scenario were defined with the intention of generating four different service-time variations: 10, 20, 30, and 60 minutes. Table 6.1 summarizes the bounds on the service times for each service action type. Note that the lower bound of the return actions was not allowed to fall below 15 minutes. This was based on the assumption that returning a container can never be quicker than 15 minutes. Similarly, in the most extreme case, the lower bounds on pick up and delivery actions were minimized at 30 minutes. In addition to the bounds, Table 6.1 provides the average and empirical standard deviation values computed for the different service types of the separate scenarios. Note that due to the lower-bound constraints on the service times, the M values of the return service times in the S_{1200}, S_{1800} cases as well as both pick-up/delivery and return M values of the S_{3600} scenario differ from the 1 hour and the $1/2$ hour values, expected by the planning methods.

The four independent-service-time scenarios S_{600}, S_{1200}, S_{1800}, and S_{3600}, of course, do not cover the infinite spectrum of possible service time variations. The exact values of 10, 20, and 30 minutes were chosen to equally sample the most probable range of service time variations. In contrast, the 1 hour case serves to display the behavior of the planning methods in extreme conditions.

6.1.3 Dependent Service-Time Uncertainty

In addition to generating service times for all actions independently, we also considered scenarios, in which service times at the same locations are always the same. To achieve this, we generated service times for every location in an instance independently according to a uniform distribution. Then, we assigned the same service times to actions happening at the same locations. The scenarios were generated according to the same variations as in the independent case (summarized in Table 6.2). This resulted in four dependent service-time variation scenarios: Sd_{600}, Sd_{1200}, Sd_{1800}, and Sd_{3600}. Just as in the case of independent service-time scenarios, Table 6.2 includes

Table 6.2: Summary of bounds on randomly generated service times for dependent service-time uncertainty scenarios, and the empirical mean M and standard deviation S of the generated datasets.

	Pick-up and Delivery Service Times			**Return Service Times**		
	Bounds [min]	M [min]	S [min]	Bounds [min]	M [min]	S [min]
Sd_{600}	[50, 70]	60	6	[20, 40]	30	6
Sd_{1200}	[40, 80]	59	12	[15, 50]	33	10
Sd_{1800}	[30, 90]	60	17	[15, 60]	37	13
Sd_{3600}	[30, 120]	75	26	[15, 90]	53	22

the empirical mean and standard deviation values computed from the generated instances. Similar to Table 6.1, the effects of the lower-bound constraints are clearly visible on the M values of the Sd_{1200}, Sd_{1800}, and Sd_{3600} scenarios.

The motivation for the scenarios with dependent service times stems from practice. For example, if the cause of a delay at a sea terminal is a faulty crane, then it will influence all trucks visiting that terminal. The situation is similar at customer sites, where a sudden shortage of labour force might hinder truck handling for the whole day.

In independent (S_*) and dependent (Sd_*) scenarios the lower index of the scenario shows the service-time variations that are in effect during the simulation. We omit the S_*/Sd_* notation in other scenarios, where the service times that are used for planning (one hour pick up, one hour delivery, half an hour return) are also used for simulation.

6.1.4 Truck-Breakdown Uncertainty

To model truck-breakdown uncertainty, we assigned breakdown times to randomly selected trucks in the static (R_0) instances. The time points in the instances when trucks break down were generated as a uniform random number between 6AM and 6PM.

With this method, we prepared three different truck-breakdown scenarios, with three (B_3), five (B_5), and ten (B_{10}) trucks breaking down out of 40 each day. In other scenarios (R_*, S_*, Sd_*), where no trucks are set to break down during the day, we omit the B_* notation.

6.1.5 Mixed-Uncertainty Scenarios

In practice, a LSP is likely to experience new jobs arriving during the day, variations in service times, or some of the trucks encountering technical problems, within the same day. Therefore, we defined problem instances, in which multiple sources of uncertainty are present. To perform a systematic search on the parameter space, we followed a two-level factorial experiment design.

In a two-level factorial design, the researcher takes a minimum and a maximum value for each input variable and runs 2^k experiments with all possible combinations, where k is the number of input parameters. In our case, the parameters we would like to vary are the parameters of the uncertainty types. Therefore the dimensions of our parameter space are the uncertainty types – which includes both the independent and dependent service times. Accordingly, we

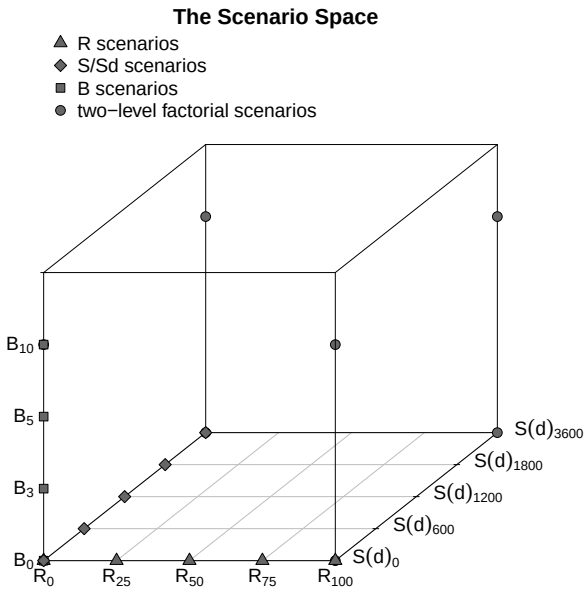


Figure 6.3: Designated scenarios in the input parameter space.

have to select the minimal and maximal parameter values for each uncertainty type. In our case, the minimal value for each dimension is the zero uncertainty value, i.e. the case when the given uncertainty is not present in the instances. The maximal uncertainty in the release-time uncertainty case is represented by R_{100}, in the service-time uncertainty cases S_{3600} and Sd_{3600}, and in the truck-breakdown case B_{10} (See Figure 6.3). Table 6.3 summarizes the scenarios that were simulated to test the planning methods. It contains the full ranges of the pure scenarios as well as the ones that are prescribed by the two level factorial design. Note that the pure scenarios that are also part of the factorial design are not repeated in the bottom half of the table.

Before we proceed to the results of the two approaches in the scenarios defined above, we first present a comparison of different combinations of the agent heuristics. Our goal in the next section is to see how different combinations of the heuristics perform, and to select some of the combinations for the comparison with the on-line optimization approach.

6.2 Comparison of Agent Heuristics

To understand how the different algorithms used in the agent approach improve quality, we chose to test them on the R_0 and R_{100} scenarios. The reason for this is a desire for compatibility and comparability with previous research. The R_0 instances belong to the class of static instances that are the basis of any performance tests in vehicle-routing literature. Similarly, the R_{100} instances belong to the class of dynamic instances that have an equally important role in the dynamic-vehicle-routing literature.

As described in Section 4.1.1, the agent approach consists of four different algorithms, namely the simple insertion method, its extension with substitutions, and two improvement methods: relocation and exchange. The simple insertion method provides an initial plan that is improved by the three other algorithms. Because of this, any combination of the methods handle job arrival uncertainty the same way as the insertion method – by inserting the new order into

Table 6.3: Uncertainty scenarios.

Pure scenarios	
R_0, R_{25}, R_{50}, R_{75}, R_{100}	different % of jobs with release time $>$ 6AM
S_{600}, S_{1200}, S_{1800}, S_{3600}	10, 20, 30, and 60 minutes independent service-time variations
Sd_{600}, Sd_{1200}, Sd_{1800}, Sd_{3600}	10, 20, 30, and 60 minutes dependent service-time variations
B_3, B_5, B_{10}	3, 5, 10 truck breakdowns per day
Mixed scenarios	
$R_{100}S_{3600}$, $R_{100}Sd_{3600}$	100 % dynamic jobs with 60 minutes independent and dependent service-time variations
$R_{100}B_{10}$	100 % dynamic jobs with 10 truck breakdowns per day
$S_{3600}B_{10}$, $Sd_{3600}B_{10}$	60 minutes independent and dependent service-time variations with 10 trucks breaking down per day
$R_{100}S_{3600}B_{10}$, $R_{100}Sd_{3600}B_{10}$	100 % dynamic jobs, 10 truck breakdowns per day, with 60 minutes independent and dependent service-time variations

the plan of one of the trucks.

To compare the different combinations of agent methods, we simulated all 33 instances of the static (R_0) and the dynamic (R_{100}) scenarios using the following combinations of methods.

1. Insertion only – denoted as *ins*,
2. insertion with relocation – denoted as *ins+rel*,
3. insertion with exchange – denoted as *ins+ex*,
4. insertion with relocation and exchange – denoted as *ins+rel+ex*,
5. insertion with substitution – denoted as *sub*,
6. insertion with substitution and relocation – denoted as *sub+rel*,
7. insertion with substitution and exchange – denoted as *sub+ex*,
8. insertion with substitution, relocation, and exchange – denoted as *sub+rel+ex*.

The upper two diagrams of Figure 6.4 depict the cost values computed from the execution traces in the R_0 and R_{100} scenarios as produced by the nine agent methods plus the on-line optimization. The bottom chart reveals the measured robustness of the methods.

Looking at the results of the static (R_0) instances (the top chart in Figure 6.4), we can see how the additional agent methods improve the performance of the basic insertion method. Interestingly, our substitution method stands out from the additional methods, since the relocation and/or the exchange methods cannot further improve the performance of substitution. Nevertheless, in the static scenario the on-line optimization dominates all of the agent methods.

In the dynamic instances (the middle chart in Figure 6.4), however, the on-line optimization no longer dominates the agent methods. Of the different agent-method combinations, the ones that include the substitution method perform better, although the differences are rather small. Comparing the upper two diagrams, we can see that the agent-method combinations including substitution perform better, in both static and dynamic scenarios. The addition of further improvement methods (relocation, exchange) over substitution yields better results in the static case, but not in the uncertain scenarios.

The bottom diagram in Figure 6.4 compares the methods using the robustness measure, as defined in Section 3.4, which is the ratio of the cost-based performance in the static and the dynamic instances, maximized at one, and averaged over the 33 instances. The closer these averages are to one, the lower the effect of the dynamic order arrivals are, on the given method. Therefore, it is more resistant to new order arrivals, and on the whole, more robust. The results at the bottom of Figure 6.4 tell us that all agent methods are more robust against new order arrivals than on-line optimization. The differences between the agent methods are again small, and interestingly, our substitution method does not have the same competitive edge as in the cost-value comparisons. In summary, these findings justify hypothesis 6.1, because while the substitution heuristic could improve the cost effectiveness of the agent-based solution method in the static scenario, it did not have a negative effect on the costs in the dynamic scenario.

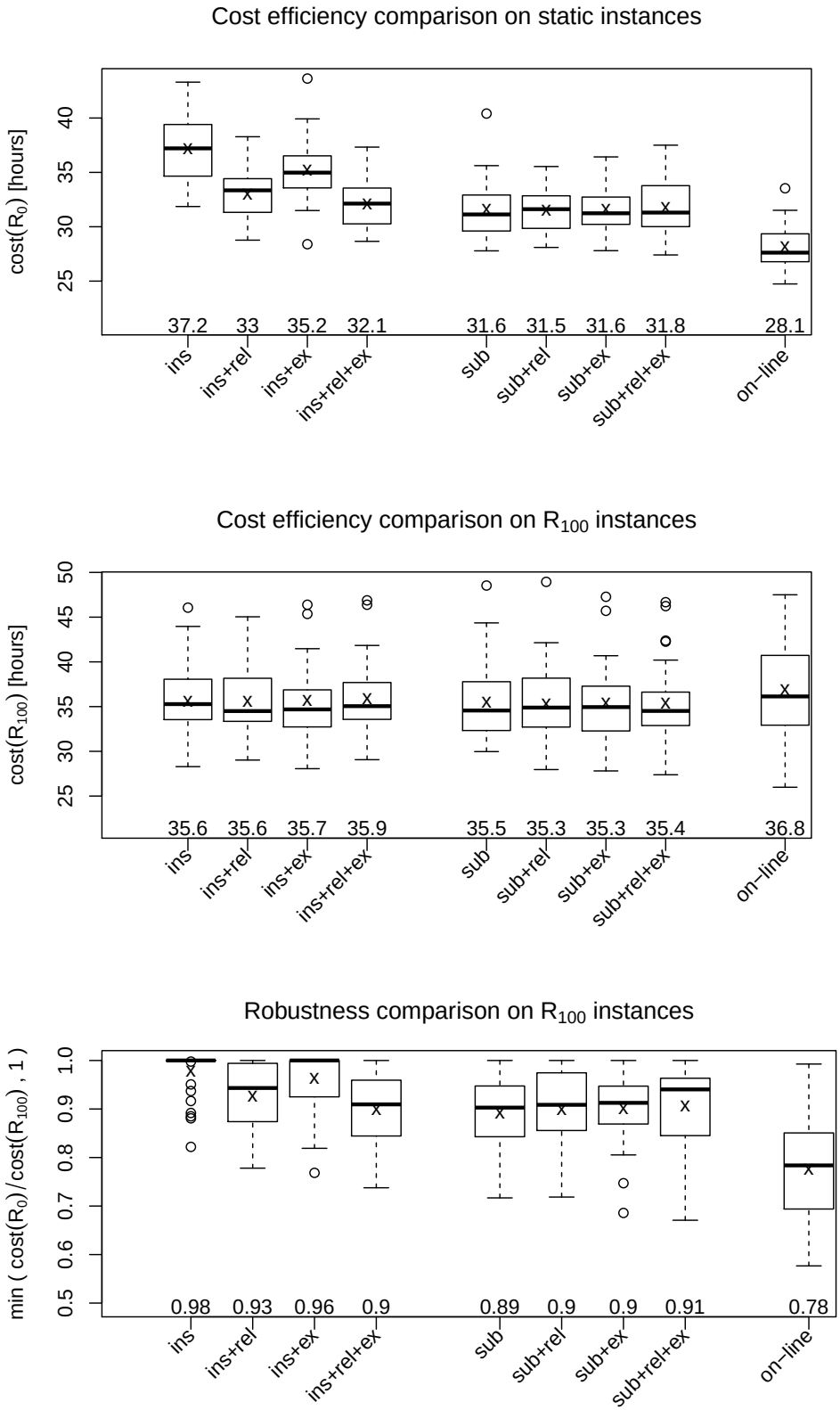


Figure 6.4: Comparison of quality and robustness of the agent procedures and the MIP optimization on R0 and R100 instances.

We now need to select which agent methods should compete against the on-line optimization across the various uncertainty scenarios. Considering both the cost-value and robustness performances, we choose to run the *ins+rel+ex* method, because it was among the best in terms of robustness, and because it represents the state of the art. Additionally, we add *sub+rel+ex* to the comparison, because it was one of the best in terms of cost efficiency, and because of the symmetry in using both the relocation and exchange improvement methods in both (*ins+rel+ex*, *sub+rel+ex*) agent methods. Given this selection of agent heuristics, the next section provides the results and analysis of the second simulation phase.

6.3 Comparison of the Centralized and Distributed Approaches

In the following, we present our findings about the performance of the algorithms described in Chapter 4. These findings include the classical (cost) measure, as well as the robustness measure defined in Chapter 3. In presenting the results, we apply the following scheme in all subsections. First, we recap how the methods handle the unforeseen events that occur due to the uncertainties present in the given scenarios. Then we provide the numerical results on the methods' cost efficiency and robustness. Both types of results are represented using box-and-whisker diagrams, where the horizontal lines of the boxes represent the lower quartile, the median, and the upper quartile of the data, and the whiskers extend to the most extreme data points that are no more than 1.5 times the interquartile range from the box. In addition, in each box-plot an 'X' denotes the location of the sample mean, and the corresponding numerical value is also printed below the box-plot. On the cost-efficiency charts, $cost(X)$ denotes the total cost of instance X, which is the sum of the empty travel time, the rejection penalty, and the lateness measured in hours as computed from the execution traces. To decide whether the cost or robustness results of two methods are statistically different, when it is not obvious from the figures, we employ t-tests. In particular, when one of the methods is better than another one at lower levels of uncertainty, but is worse than the other at higher levels, we use t-tests to clarify the level of uncertainty, at which the change happens. In the t-tests, we consider a level of $p < 0.05$ statistically significant.

6.3.1 Release-Time Uncertainty

Release-time uncertainty scenarios extend the static transportation instances by allowing new transportation requests to arrive during the execution of already planned orders. As described in Section 6.1, we defined five different release-time scenarios with 0%, 25%, 50%, 75%, and 100% dynamic orders (not available in the morning) in the instances. In this section, we compare the results of two agent-based planning methods (*ins+rel+ex*, *sub+rel+ex*), the on-line optimization method, and the *a posteriori* optimal results. How the two agent-based methods handle new orders was discussed in the previous section. The on-line optimization incorporates new orders by simply adding them to the immediate next MIP formulated after their arrival, thus the next solution generated by the solver already contains plans to transport the new orders. The *a posteriori* optimization solves only one MIP to optimality, where all orders are considered with their true arrival time. This reflects an end-of-day analysis showing how good a planner could

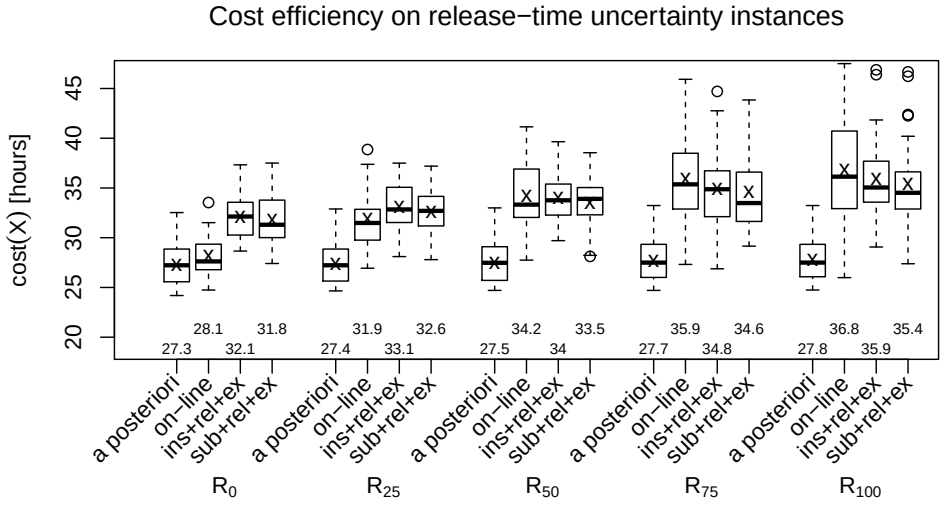


Figure 6.5: Comparison of method quality on release-time scenario instances.

be, if she had known the exact arrival times of all jobs.

6

Cost Efficiency

Not surprisingly, the *a posteriori* results dominate all of the other methods across the full range of release-time uncertainty instances. It is interesting to observe that these results hardly change as the percentage of dynamic jobs grow. This means that dynamic instances entail only slightly more empty travel per se, because of slightly tighter time constraints. The performance loss of other methods in more dynamic instances can be mainly attributed to a lack of knowledge about future jobs.

The on-line optimization is very close to the *a posteriori* optimal in the R_0 scenario. The slight difference is caused by the real-time nature of the simulation. While it takes some time for the optimizer to come up with an initial solution, the time in the simulator ticks on, causing some parts of the initial plan to be outdated by the time the on-line planner sends them for execution. In contrast to the small gap in the R_0 scenario, as the percentage of dynamic orders increases, the amount of empty-travel time and rejection cost increase monotonically, and the gap between the *a posteriori* optimal results continues to widen. The trend is the same for the two agent-based methods, but the gap between methods across scenarios is dampened. Both produce considerably worse results in R_0 than the on-line optimization, but an increase in the number of dynamic orders does not significantly influence these methods. In fact, the agent methods and the on-line optimization are nearly the same at R_{50}, and for more dynamic instances the agent methods outperform the on-line optimization. Of the two agent methods, the substitution-based method consistently produces better results than the insertion-based method, although the difference between the two is small, and does not change as the amount of dynamic orders increases.

To see whether the differences between the on-line optimization and the better performing agent method (*sub+rel+ex*) are statistically significant, we performed paired two-sample Welch t-tests for all R_* scenarios. We used paired tests because the results of the two methods were generated from the same input, thus the results are not statistically independent, and paired

Table 6.4: Paired two-sample Welch t-tests of R_{25}, R_{50}, R_{75}, and R_{100} results for the on-line optimization and the sub+rel+ex *agent method (null hypothesis: the means of the results are equal).*

	R_{25}	R_{50}	R_{75}	R_{100}
t-value	−1.98	1.43	1.91	2.44
degree of freedom	32.00	32.00	32.00	32.00
p-value	0.06	0.16	0.07	0.02
difference of means	−0.73	0.72	1.26	1.48

tests are better in filtering out the effect of input noise. Furthermore, since the variance of the two samples is not equal, we had to use Welch tests. Because the difference of the agent methods and the on-line optimization is obviously big in R_0, we omitted this scenario, and tested only the other scenarios. An assumption of the t-test is that the samples follow a normal distribution. This is assessed by the normal quantile plots in Figure B.1 (in Appendix B). The close linearity of the points suggests that all samples are normally distributed. The results of the tests, summarized in Table 6.4, reveal that the means of the on-line optimization and the *sub+rel+ex* agent method differ significantly in all but the R_{50} case. That is, on the static end of the range, on-line optimization is significantly better than the *sub+rel+ex* method; while on the dynamic end of the range, the agent method preforms significantly better.

Robustness

After examining the cost-value comparisons of the on-line optimization and the agent methods, we turn to the robustness results of these planning methods (see Figure 6.6). Note that our robustness measure is one for R_0 with any method, because R_0 is the static scenario, and R_0 is the numerator of the robustness measure.

Quite interestingly, the agent methods are consistently more robust than the on-line optimization throughout the whole spectrum of release-time uncertainty instances. The agent-based approaches yield better cost results, and are less sensitive to all levels of release-time uncertainty than the on-line optimization method. Furthermore, between the two agent methods there is very little difference, they perform equally. The *a posteriori* method sores very close to one, of course. The only reason for not being exactly one is that the optimal solutions of the more dynamic instances are more costly than the optimal solutions of the static instances.

After presenting the results of the R_* scenarios, which are the standard scenarios for dynamic vehicle routing methods, we move on to the other uncertainty scenarios. The scenarios documented in the following sections are less frequently considered in the literature; thus the setting, in addition to the results themselves, can be considered a contribution.

6.3.2 Independent Service-Time Uncertainty

Independent service time scenarios, as described in Section 6.1, extend the static R_0 instances with random service times drawn independently from uniform distributions with different bounds.

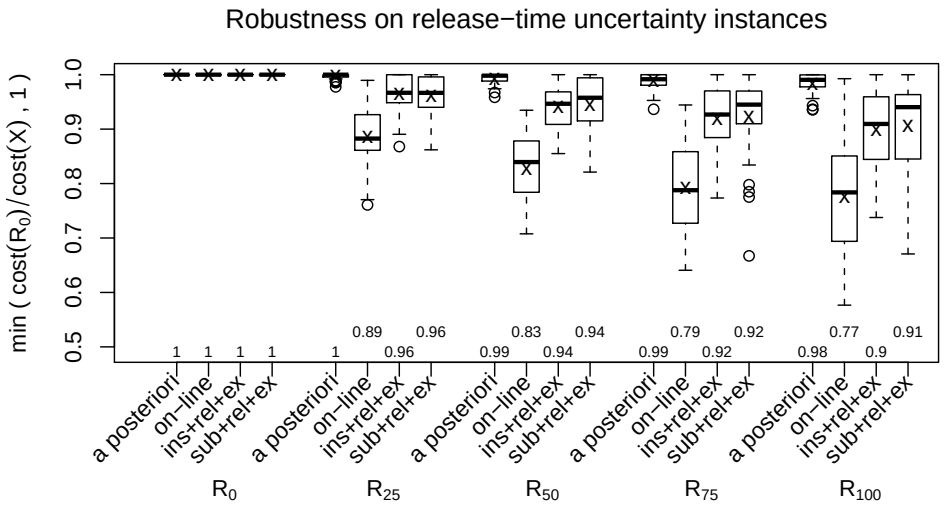


Figure 6.6: Robustness measures of the two agent-based and the on-line optimization methods for release-time scenario instances.

The randomly drawn service times are applied at the service locations (pick up, delivery, return locations) of the jobs. They are not visible to the planning methods at the planning time, instead they are applied during execution. As a result, execution will diverge from the plans by different measures, and the planning methods will have to reconsider the plans, should the gap between the plans and the execution grow too large.

In the on-line optimization method, variable service times are handled by simply shifting the containers in time. Removing already assigned containers from trucks is avoided, because we experienced heavy rejection rates when this is allowed. This can be attributed to the impossibility of adding more containers to the already optimized plans of trucks. In the agent methods however, truck agents remove containers from their plans when they realize that delivery will be late. The removed container agents try to find another truck by auctioning, thus using their standard tools. Finally, an *a posteriori* solution is not computed in this scenario, because the objective function of the methods does not allow, in principle, lateness, although it still happens due to the random service times affected during execution. If we used the same objective function for an a posteriori analysis, it would generate solutions with zero lateness and a high level of rejection, thus producing results incomparable to the on-line methods.

Cost Efficiency

The cost results of the simulation runs for the independent service-time scenarios are depicted in Figure 6.7. Note that similar to the previous figures, the leftmost scenario, R_0, is the static case. Figure 6.7 reveals that the transportation cost induced by each method increases as the uncertainty in service times increase. While in the low uncertainty cases the on-line optimization performs better, the agent methods perform better from the S_{1800} scenario on.

Although it is not visible on these figures, the on-line optimization method produces increasingly more late deliveries than the agent methods do, as service-time uncertainty increases. At the extreme end, where the service-time variation is one hour, the difference between the on-line

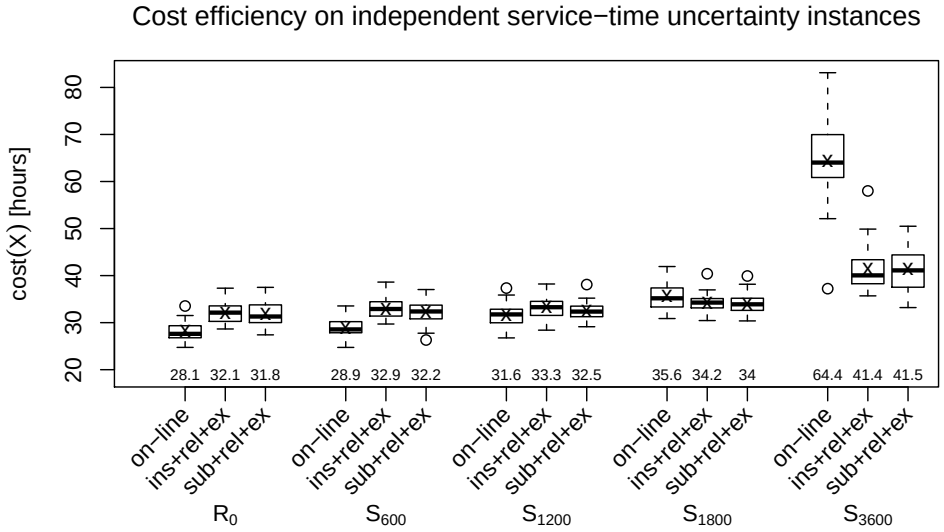


Figure 6.7: Cost comparison in independent service-time scenario instances.

Table 6.5: Paired two-sample Welch t-tests of S_{600}, S_{1200}, S_{1800}, and S_{3600} results for the on-line optimization and the sub+rel+ex *agent method (null hypothesis: the means of the results are equal).*

	S_{600}	S_{1200}	S_{1800}	S_{3600}
t-value	−10.82	−4.13	2.97	14.20
degree of freedom	32.00	32.00	32.00	32.00
p-value	0.00	0.00	0.01	0.00
difference of means	−3.30	−0.90	1.58	22.96

optimization and the agent method is huge. This is primarily caused by the inability of the on-line optimization to reschedule already assigned jobs, which increases the probability that subsequent jobs miss their time windows as a result of earlier service-time delays. Furthermore, the highly optimized plans have very little room to adjust the ordering of jobs in case one of them is late. An interesting difference between the agent methods and the on-line optimization is that the agents find a better balance of rejected and late jobs (although this balance is not explicitly encoded in them), while the on-line optimization is unable to reject jobs due to the reasons detailed above.

Table 6.5 summarizes the paired two-sample Welch t-test results that we performed to see whether the cost values of the on-line optimization and the *sub+rel+ex* agent method are significantly different. The assumption of the test, that the samples are from a normal distribution is justified by the normal-quantile plots in Figure B.2 (in Appendix B). The t-tests reveal that the means of the cost values for the two methods differ significantly in all scenarios. In the S_{600} and S_{1200} cases, the on-line optimization is better, while in the S_{1800} and S_{3600} cases the agent methods have less transportation costs.

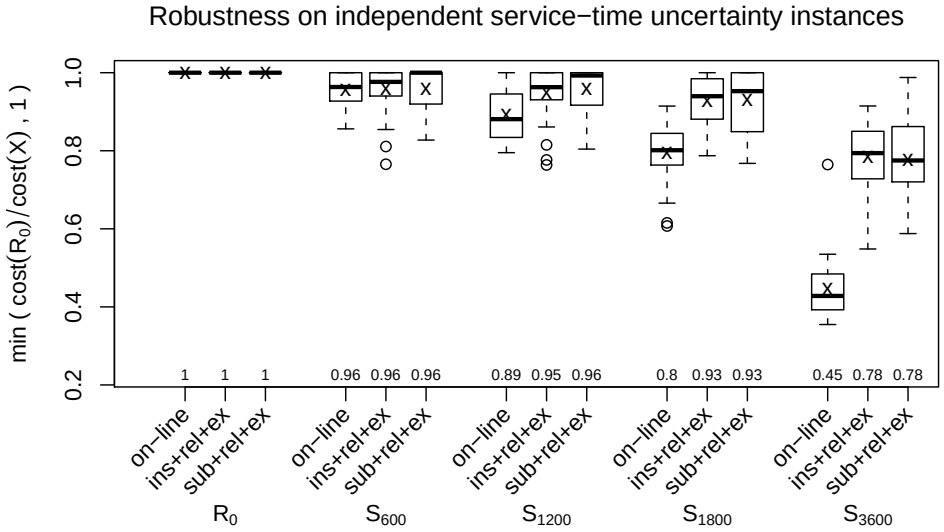


Figure 6.8: Robustness comparison in independent service-time scenario instances.

Robustness

The robustness results depicted in Figure 6.8 are slightly different from the robustness values found in the release-time uncertainty instances. While the trends are the same, in the S_{600} scenario (with lower uncertainty), the on-line optimization and the agent methods have an equal robustness value. In the higher uncertainty cases, just as in the release-time uncertainty scenarios, the quality degradation of the on-line optimization method far exceeds the degradation of the agent methods. It is interesting to note that in the S_{1200} scenario, the on-line optimization performs better in terms of cost, but is less robust than the agent methods. This indicates that the agent methods should outperform the on-line optimization method cost-wise, which is what happens in the more uncertain scenarios.

In the independent service-time scenarios of this section, service times for all container actions (pick up, delivery, return) were generated independently. In the next section, we discuss a variation of these service-time uncertainty scenarios, in which the generated service times are not independent.

6.3.3 Dependent Service-Time Uncertainty

Dependent service-time scenarios are very similar to the independent service-time scenarios, the difference being that, here, random service times are drawn independently for every location in the problem, instead of for every action. The result is that container orders that have the same locations as their pick-up, delivery, or return location will have the same service times at those locations. The handling of dependent service-time uncertainty happens the same way as independent service-time uncertainty.

Cost Efficiency

Figure 6.9 depicts the cost results in dependent service-time scenarios. This figure resembles the previous one in that the same trend and almost the same numbers can be observed. All methods

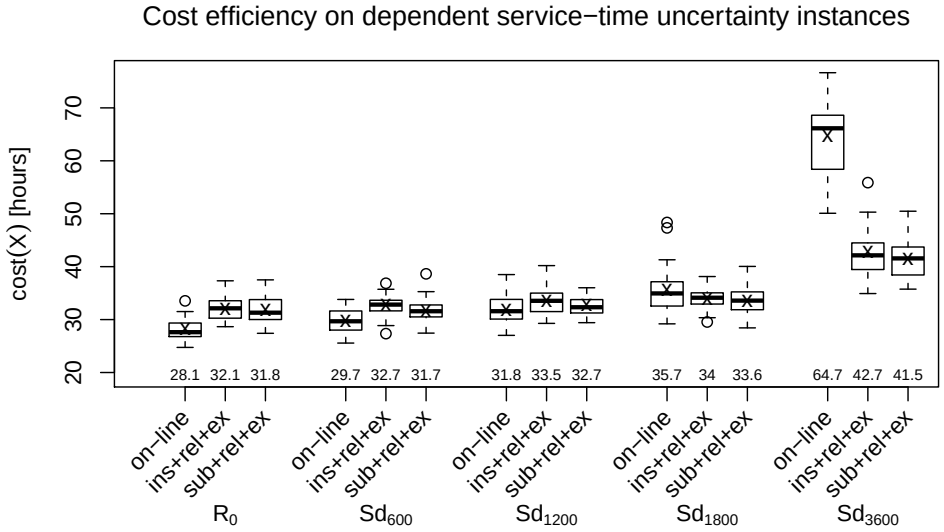


Figure 6.9: Cost comparison in dependent service-time scenario instances.

Table 6.6: Paired two-sample Welch t-tests of Sd_{600}, Sd_{1200}, Sd_{1800}, and Sd_{3600} results for the on-line optimization and the sub+rel+ex *agent method (null hypothesis: the means of the results are equal).*

	Sd_{600}	Sd_{1200}	Sd_{1800}	Sd_{3600}
t-value	−8.55	−2.24	3.27	18.27
degree of freedom	32.00	32.00	32.00	32.00
p-value	0.00	0.03	0.00	0.00
difference of means	−1.99	−0.93	2.10	23.21

react with increasing cost values as the uncertainty in the instances increases. The agent methods outperform the on-line optimization when service times vary by 30 or more minutes.

Table 6.6 summarizes the paired two-sample Welch t-test results performed to test the difference between the cost results of the on-line optimization and the *sub+rel+ex* agent method. The assumption of the test, that the samples are from a normal distribution, is justified by the normal-quantile plots in Figure B.3 (in Appendix B). The result of the tests is that the means of the cost values of the two methods differ significantly in all scenarios. In the Sd_{600} and Sd_{1200} cases, the on-line optimization is better, while in the Sd_{1800} and Sd_{3600} cases the agent methods have lower transportation costs.

Robustness

The robustness results depicted in Figure 6.10 are also very similar to the robustness values in the independent service-time uncertainty case. The agent methods are more robust in all dependent service-time uncertainty scenarios, even at low-uncertainty levels. The two agent methods are also equally robust here.

After introducing the results of the three planning methods on service-time uncertainty instances, in the next section, we turn our attention to scenarios rarely discussed in the literature:

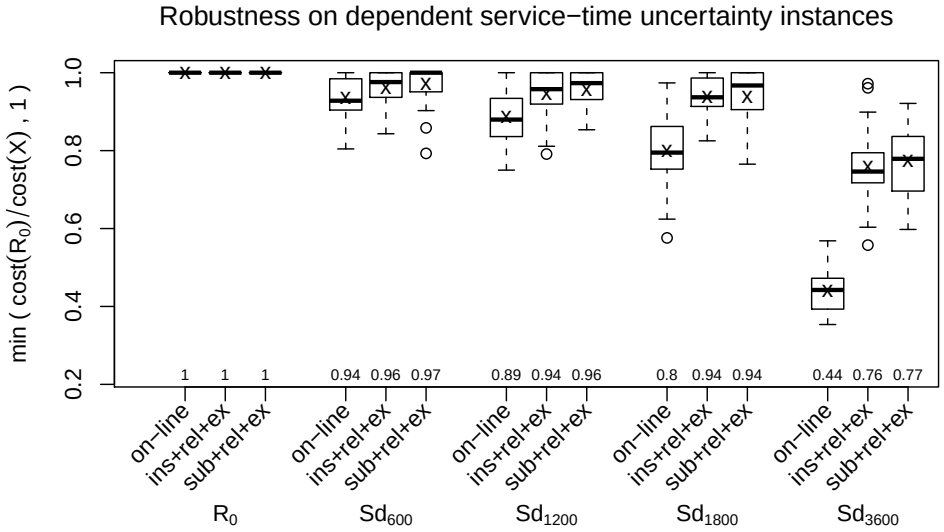


Figure 6.10: Robustness comparison in dependent service-time scenario instances.

truck-breakdown uncertainty scenarios.

6

6.3.4 Truck-Breakdown Uncertainty

Truck-breakdown uncertainty scenarios describe situations in which trucks fall out of operations on a given day, and containers assigned to the broken truck have to be transported by other trucks. As described in Section 6.1, we compare the two agent methods and the on-line optimization on three different scenarios B_3, B_5, and B_{10}, where 3, 5, and 10 trucks break down, respectively.

When a truck breaks down during the pick up of a container, the container can be picked up at the same location by another truck. If the truck breaks down on the road after the container is picked up, the container is available for other trucks to pick it up at the location where the previous truck broke down. Once the truck reaches the delivery location of a container, the container is not considered for transport by another truck, irrespective of the state of the container at the breakdown time (i.e. being delivered, on the road to the return location, or being returned). The reason for this is that the most important thing is delivering the container to the customer, and containers delivered by broken trucks can be collected on another day.

All methods consider broken trucks according to the above guidelines. When the notification of a truck breakdown arrives to the planning methods, the agents re-auction the containers that are not delivered yet, while the on-line optimization method includes the yet undelivered containers into the next MIP. Thereby both methods try their best to reschedule the containers which were assigned to the broken truck.

Cost Efficiency

Figure 6.11 shows the cost results of the three methods in the B_3, B_5, and B_{10} scenarios. The trend again is that the more uncertainty there is in the instances, the worse cost values the methods produce. In this case however, the on-line optimization method retains its advantage through the whole range of test scenarios. Even when 25% of the trucks break down, the on-line optimization keeps an approximately 4 hour (which is about 9-10%) advantage compared to

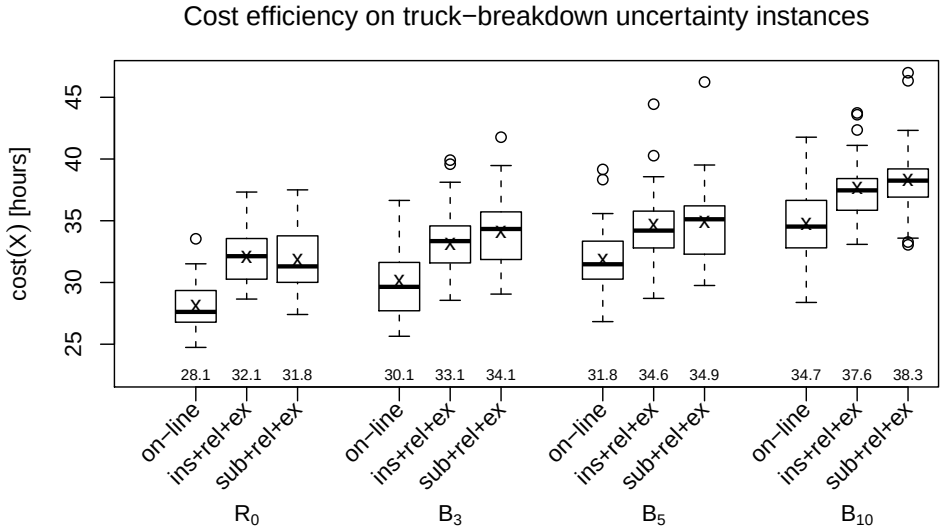


Figure 6.11: Cost comparison in truck-breakdown scenario instances.

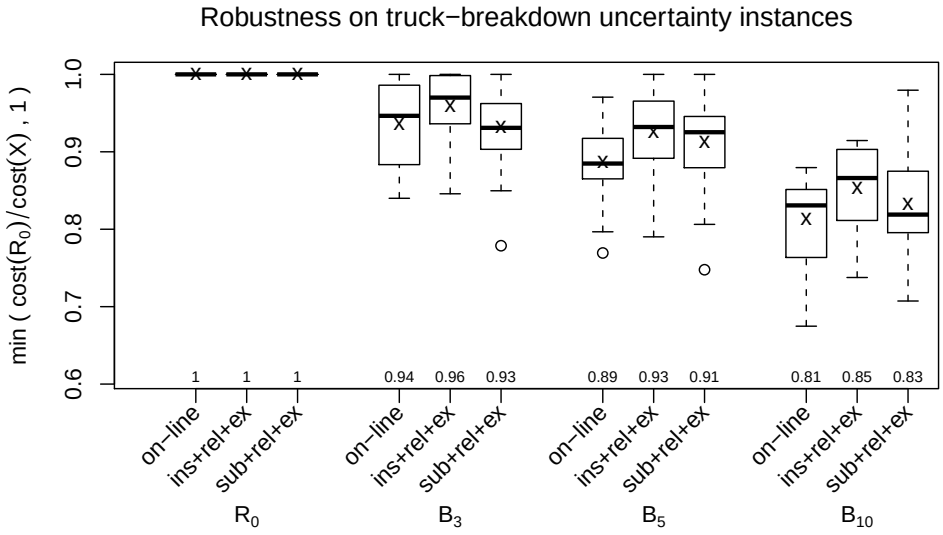


Figure 6.12: Robustness comparison in truck-breakdown scenario instances.

the agent methods. This uncertainty type is different from the previously seen ones in that the agent heuristics cannot produce better cost results than the on-line optimization at any level of truck-breakdown uncertainty; they do not even converge to the on-line optimization. From the perspective of cost efficiency, truck-breakdown uncertainty has the same effect on the agent methods as on the on-line optimization.

Robustness

Interestingly, the robustness values in the truck-breakdown scenarios do not follow the previously seen pattern that the method with the least cost in the uncertain cases is the most robust. Even though the on-line optimization performs the best cost-wise throughout the B_* scenarios, it is not more robust than the agent methods in any of the scenarios.

The B_* scenarios are unique in one more aspect. Here, the cost results are clearly different

Table 6.7: Paired two-sample Welch t-tests of B_3, B_5, and B_{10} robustness results of the on-line optimization and the two agent methods (null hypothesis: the means of the results are equal).

	ins+rel+ex			***sub+rel+ex***		
	B_3	B_5	B_{10}	B_3	B_5	B_{10}
t-value	−1.94	−3.21	−3.79	0.33	−2.13	−1.67
degree of freedom	32.00	32.00	32.00	32.00	32.00	32.00
p-value	0.06	0.00	0.00	0.75	0.04	0.11
means of difference	−0.02	−0.04	−0.04	0.00	−0.03	−0.02

for the on-line optimization and the agent methods (in contrast to earlier cases, where we needed statistical tests to tell the difference), but their robustness values are close. Therefore, instead of testing the cost results for statistical difference, we now test the robustness of the two agent methods and the on-line optimization method. Again, we apply paired two-sample Welch t-tests. The normality of the samples are justified by the normal-quantile plots in Figure B.4 (in Appendix B), and the results of the tests are summarized in Table 6.7. The p-values obtained show that, in the case of the *ins+rel+ex* method, we can reject the null-hypothesis that the robustness values are equal to the on-line optimization, with the exception of the B_3 scenario. That is, for higher levels of truck-breakdown uncertainty, the *ins+rel+ex* method is more robust than the on-line optimization. The same does not hold for the *sub+rel+ex* method, which is significantly different (with 95% significance) in only the B_5 scenario, in which it is more robust than the on-line optimization, although only by a marginal 2%. In the rest of the cases, the difference of the mean values of the *sub+rel+ex* and the on-line optimization methods are not significant (again, at a 95% significance level). The conclusion that can be drawn from this is that having close robustness values predicts that the cost-based performance of the methods change in parallel as uncertainty grows. And indeed, this is what we see in Figure 6.11.

This far we presented the results of pure-uncertainty scenarios only. The general conclusion is that under any type of uncertainty, as the level of uncertainty increases, the transportation costs incurred by the methods increase, and the robustness values of all methods decrease. There is however a difference in the observed pattern. In the high uncertainty cases of release-time and service-time uncertainties, the agent methods perform equally well or even better than the on-line optimization in terms of cost. This justifies Hypothesis 6.2 which says that the cost-advantage of the on-line optimization method decreases as the uncertainty in the problem increases. At the same time, in the truck-breakdown uncertainty scenarios, the on-line optimization consistently outperformed both agent methods, which falsifies Hypothesis 6.2. In terms of robustness, as predicted by Hypothesis 6.3, the agent methods are always more robust in the high uncertainty cases than the on-line optimization, except in the truck-breakdown scenarios, when only the *ins+rel+ex* agent method is better, and the *sub+rel+ex* is only not worse.

In the following section, we present the results of the combined scenarios, in which we can study how the different uncertainty types influence each other.

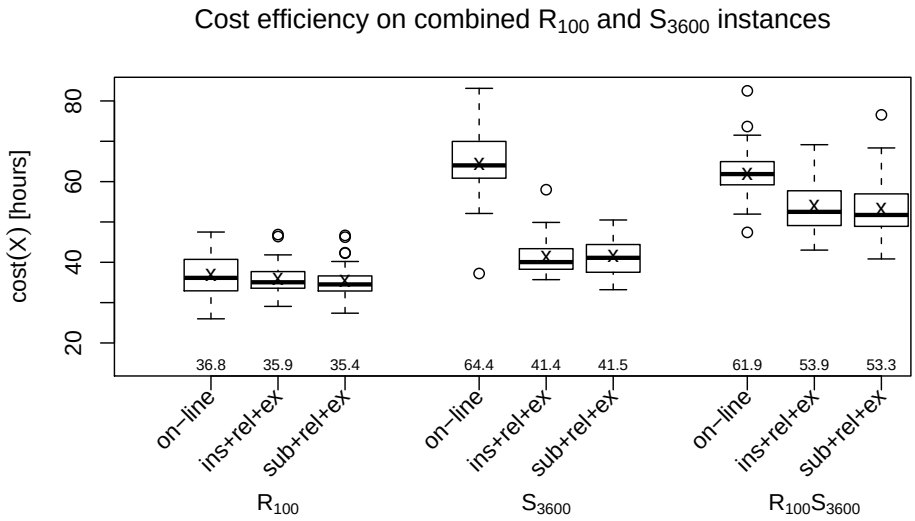


Figure 6.13: Cost results of the three methods in the R_{100}, the S_{3600}, and the $R_{100}S_{3600}$ scenarios.

6.3.5 Mixed-Uncertainty Scenarios

In the previous sections, we presented the results of the on-line optimization, and the two agent algorithms across five levels of release-time, five levels of independent and dependent service-time, and four levels of truck-breakdown uncertainty scenarios. Since, in reality, these three sources of uncertainty are usually simultaneously present, it is interesting to study cases with more than one source of uncertainty. In this section, we consider mixed-uncertainty scenarios as defined in the experimental design outlined in Section 6.1.

Release-Time and Independent Service-Time Uncertainty

We start our discussion with the $R_{100}S_{3600}$ scenario. This is composed of the most extreme release-time and independent service-time uncertainty scenarios: all orders arrive during the day, and all service times are varied by approximately one hour.

Cost Efficiency Figure 6.13 depicts the cost results of the three methods in this mixed scenario. For comparison, the results of the R_{100}, and S_{3600} pure scenarios are also included. Compared to R_{100}, all three methods performed worse in the mixed scenario. Compared to S_{3600}, however, only the agent methods suffered more, the on-line optimization did not. In fact, it seems that for the on-line optimization method having new jobs arriving during execution did not add extra difficulty. On the contrary, the dynamic arrival of jobs prompted the rejection of some jobs, and thereby yielded a lower total cost value. The two agent methods perform the same in the mixed scenario, just as in the two extreme pure scenarios.

Robustness For the robustness results a similar remark is true (see Figure 6.14). Compared to the R_{100} scenario, all three methods were less robust in the mixed scenario. Compared to the S_{3600} scenario, however, the robustness value of the on-line optimization did not decrease, only the robustness values of the agent methods. Regarding the comparison of the two agent methods, the robustness results of the *ins+rel+ex* and *sub+rel+ex* methods does not differ significantly.

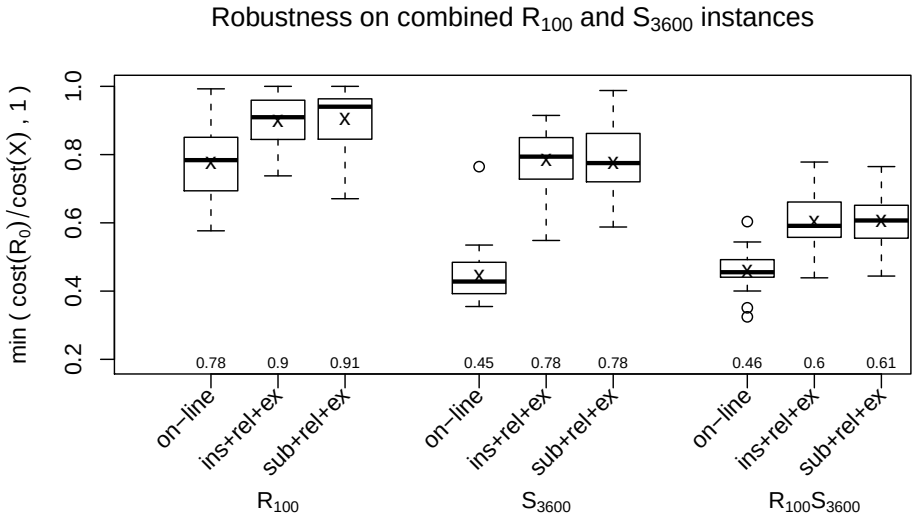


Figure 6.14: Robustness results of the three methods in the R_{100}, the S_{3600}, and the $R_{100}S_{3600}$ scenarios.

Release-Time and Dependent Service-Time Uncertainty

In the $R_{100}Sd_{3600}$ mixed scenario, the most extreme release-time and dependent service-time uncertainty scenarios are combined. The only difference with the $R_{100}S_{3600}$ scenario is that here not all actions have independent identically distributed service times. Instead, all locations in the problem have iid service times assigned to them, consequently actions at the same locations have the same service times.

Cost Efficiency Figure 6.15 reveals that the three methods perform similarly in the $R_{100}Sd_{3600}$ scenario as compared to the $R_{100}S_{3600}$ scenario. All cost results were worse in the combined scenario than in R_{100}. Compared to Sd_{3600}, however, only the agent methods realized an increase in their cost values, the on-line optimization could again reject some jobs due to dynamic job arrival, and thereby achieve a better result.

Robustness Similar to the robustness values in the $R_{100}S_{3600}$ scenario, the on-line optimization performed better in the combined case than in the Sd_{3600} case, while the agent methods proved to be less robust. Nevertheless, the agent methods still solved the problems with less cost and were more robust against the combined uncertainties as compared to the on-line optimization.

Release-Time and Truck-Breakdown Uncertainty

The last uncertainty type that the R_* scenarios can be paired with are the truck-breakdown scenarios. The combined effect of new orders arriving and trucks disappearing during execution is discussed in the following paragraphs.

Cost Efficiency Comparing the two pure scenarios R_{100} and B_{10} in Figure 6.17, the on-line optimization performed better in B_{10} than in R_{100}. Also, in B_{10} it was better than the agent methods, while in R_{100} it was worse. At the same time, the agent methods were the opposite;

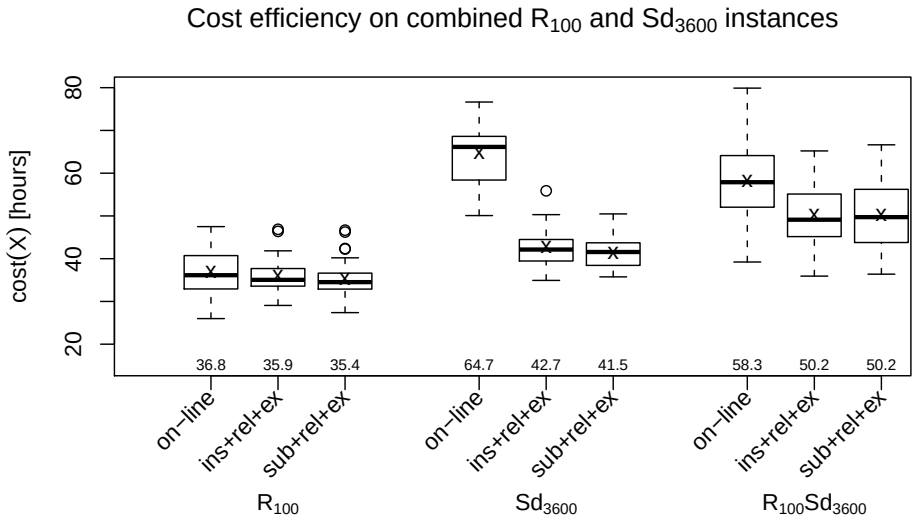


Figure 6.15: Cost results of the three methods in the R_{100}, the Sd_{3600}, and the $R_{100}Sd_{3600}$ scenarios.

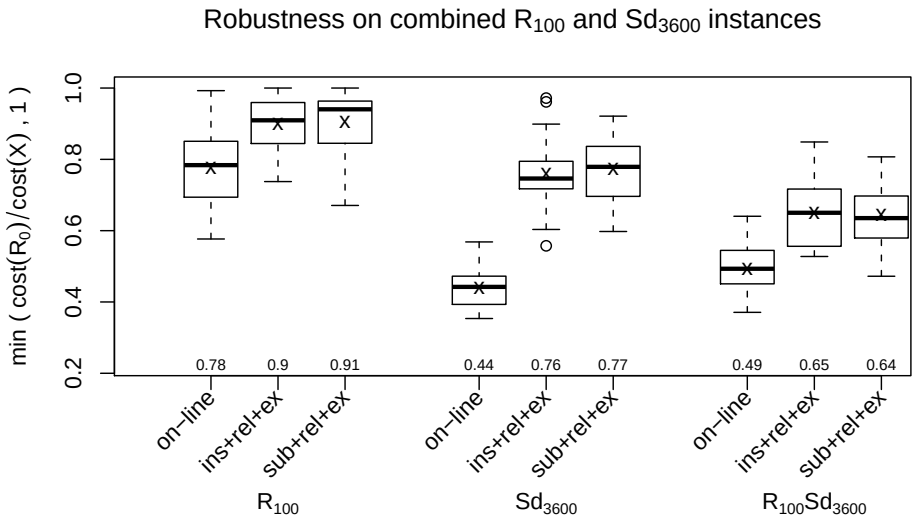


Figure 6.16: Robustness results of the three methods in the R_{100}, the Sd_{3600}, and the $R_{100}Sd_{3600}$ scenarios.

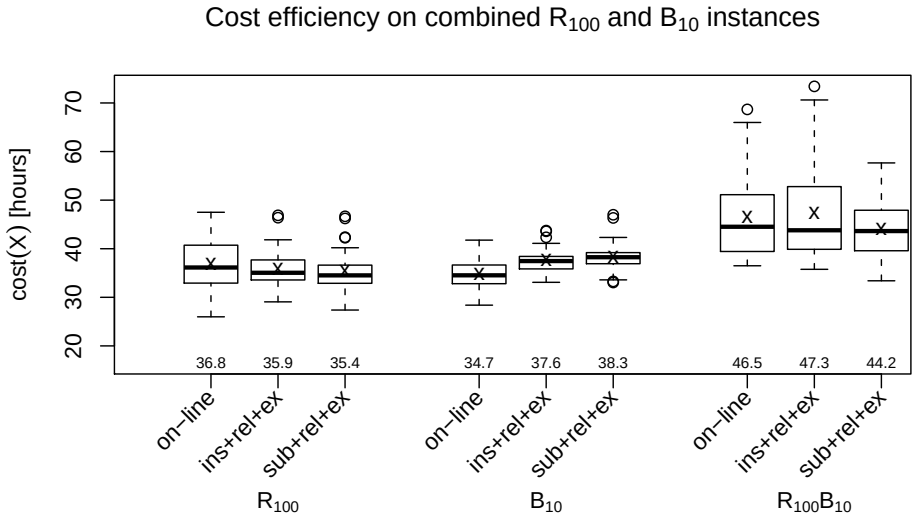


Figure 6.17: Cost results of the three methods in the R_{100}, the B_{10}, and the $R_{100}B_{10}$ scenarios.

they had a better performance in R_{100} than in B_{10}. In contrast to this alternating behavior in the pure scenarios, in the combined $R_{100}B_{10}$ scenario all three methods performed uniformly, and worse than in any of the pure scenarios. They all had higher costs.

Robustness The robustness results, as depicted in Figure 6.18, are similar in that the on-line optimization is more robust in B_{10} than in R_{100}, and for the agent methods it is the other way around. In this measure, however, the agent methods are consistently better than the on-line optimization in the pure scenarios, just as in the combined scenario.

Independent Service-Time and Truck-Breakdown Uncertainty

When we assume that all orders are static (i.e. available in the morning), we can combine the service-time and truck-breakdown uncertainty scenarios.

Cost Efficiency In Figure 6.19, the cost results in the S_{3600}, B_{10}, and $S_{3600}B_{10}$ scenarios are presented. We can observe an increase of costs for the agent methods in the mixed scenario, while the on-line optimization performed equal to the S_{3600} case. Even though the agent methods suffered a substantial performance loss in the combined scenario, they could still preserve their advantage and stay below the cost results of the on-line optimization.

Robustness Similar to the cost results, the robustness results (in Figure 6.20) of the agent methods are worst in the combined cases, while the on-line optimization stayed at the same level as in S_{3600}. Even though the agent methods lost 16-17% of their robustness value, they were still more robust than the on-line optimization. The two agent methods performed almost identically in this mixed scenario.

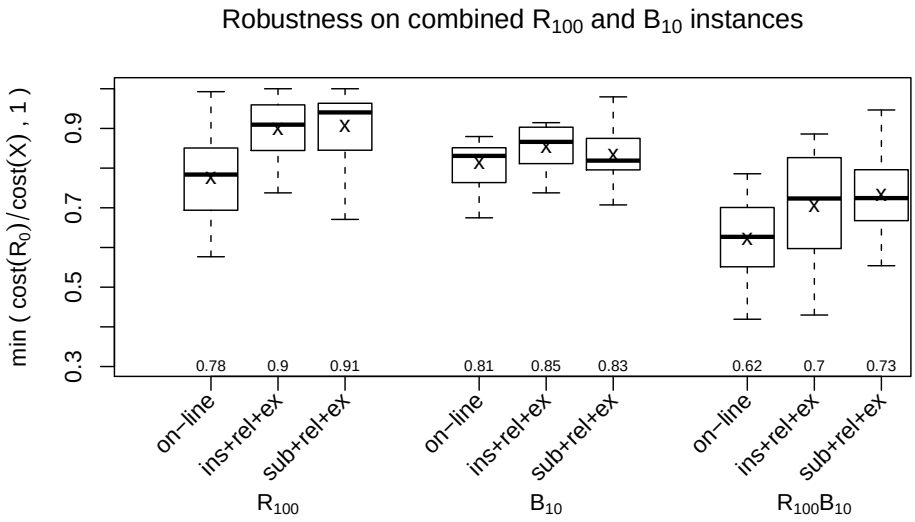


Figure 6.18: Robustness results of the three methods in the R_{100}, the B_{10}, and the $R_{100}B_{10}$ scenarios.

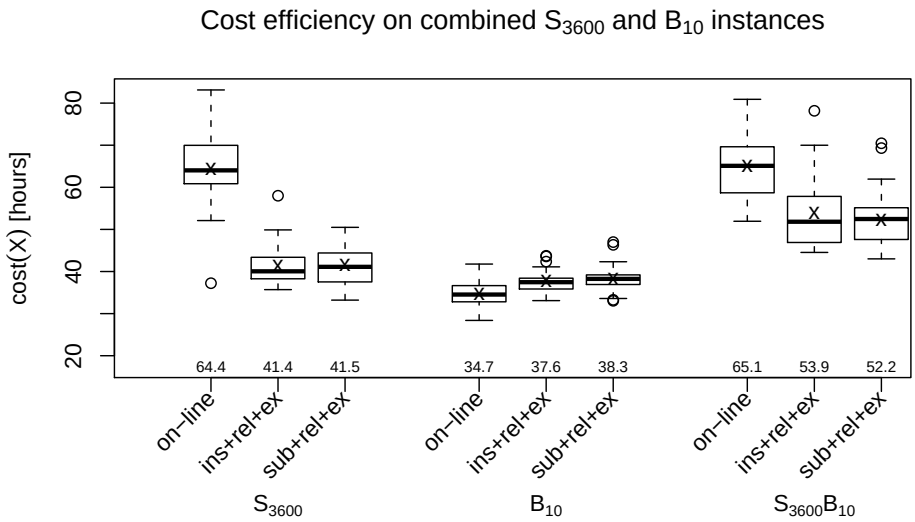


Figure 6.19: Cost results of the three methods in the S_{3600}, the B_{10}, and the $S_{3600}B_{10}$ scenarios.

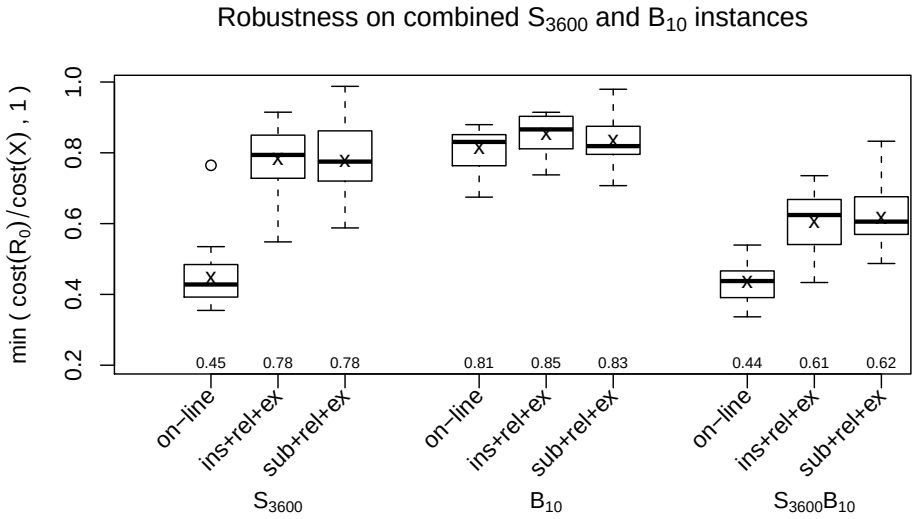


Figure 6.20: Robustness results of the three methods in the S_{3600}, the B_{10}, and the $S_{3600}B_{10}$ scenarios.

Dependent Service-Time and Truck-Breakdown Uncertainty

The combination of dependent service-time and truck-breakdown uncertainties yields problem instances similar to the mixed scenarios discussed in the previous section. Accordingly, the results are also similar.

Cost Efficiency As highlighted in Figure 6.21, the cost-based performance of the on-line optimization in the combined case equalled that of the Sd_{3600} case, while the results of the agent methods increased in cost. Even so, the agent methods still performed better than the on-line optimization, and quite similar to each other.

Robustness The robustness results in the $Sd_{3600}B_{10}$ mixed scenario are similar to the results of the $S_{3600}B_{10}$ scenario (see Figure 6.22). The robustness of the on-line optimization was equal in the combined and in the Sd_{3600} scenarios, while the agent methods were less robust in the combined-uncertainty case than in any of the pure-uncertainty cases. Nevertheless, the agent methods were more robust than the on-line optimization in all three scenarios.

Release-Time, Service-Time, and Truck-breakdown Uncertainty

In the corner of our parameter space farthest from the static instances is the maximum-uncertainty scenario. It models a catastrophic day for the planners: a day in which all three types of uncertainties are present simultaneously, and at the highest uncertainty settings.

Cost Efficiency Figure 6.23 depicts the cost-based performance of the planning methods in the $R_{100}S_{3600}B_{10}$ and $R_{100}Sd_{3600}B_{10}$ uncertainty scenarios, as well as the results in the previously shown mixed scenarios with only two sources of uncertainty. The trends already seen continue in this mixed uncertainty scenario. The total cost increased for all three methods as all of the different sources of uncertainty were combined. In all mixed scenarios, the agent methods had

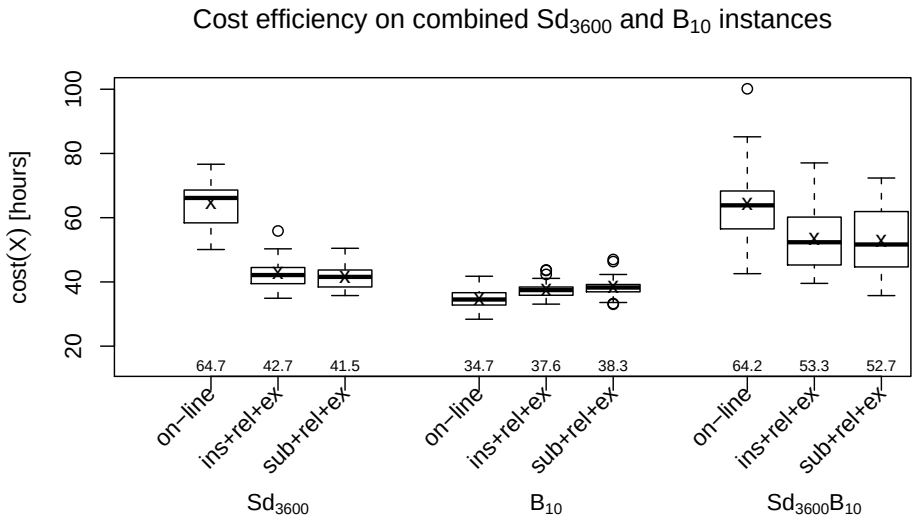


Figure 6.21: Cost results of the three methods in the Sd_{3600}, the B_{10}, and the $Sd_{3600}B_{10}$ scenarios.

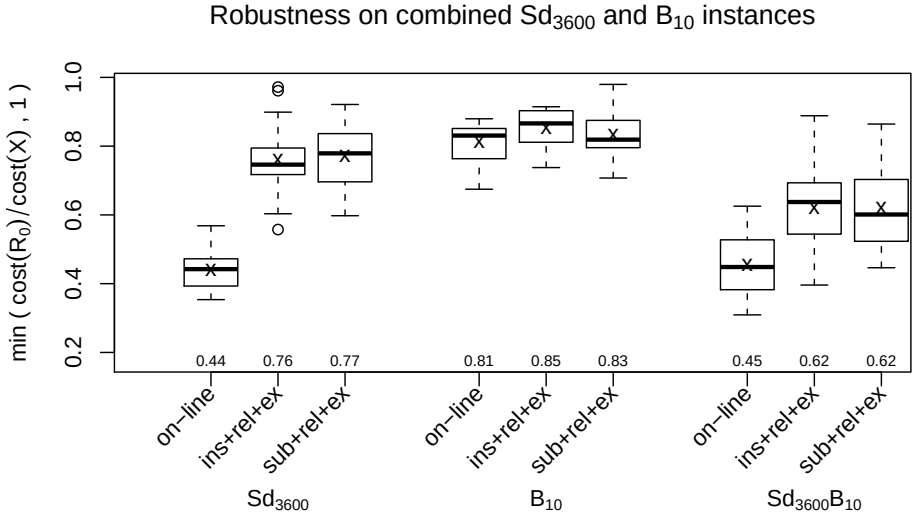


Figure 6.22: Robustness results of the three methods in the Sd_{3600}, the B_{10}, and the $Sd_{3600}B_{10}$ scenarios.

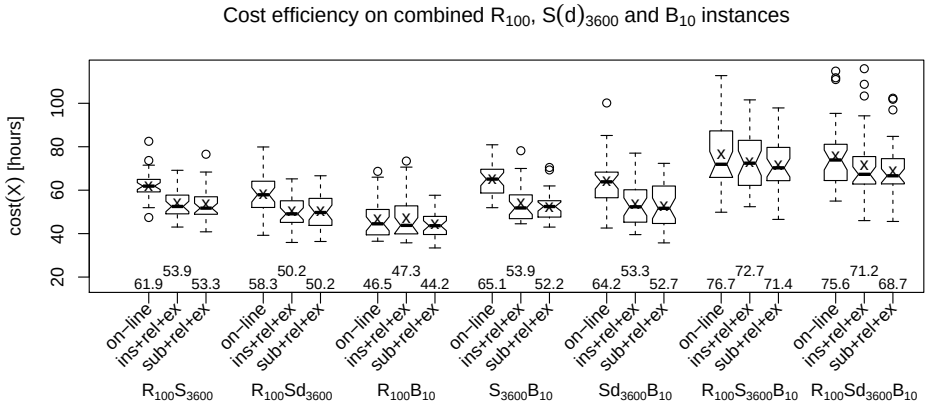


Figure 6.23: Cost results of the three methods in the different combinations of mixed-uncertainty scenarios.

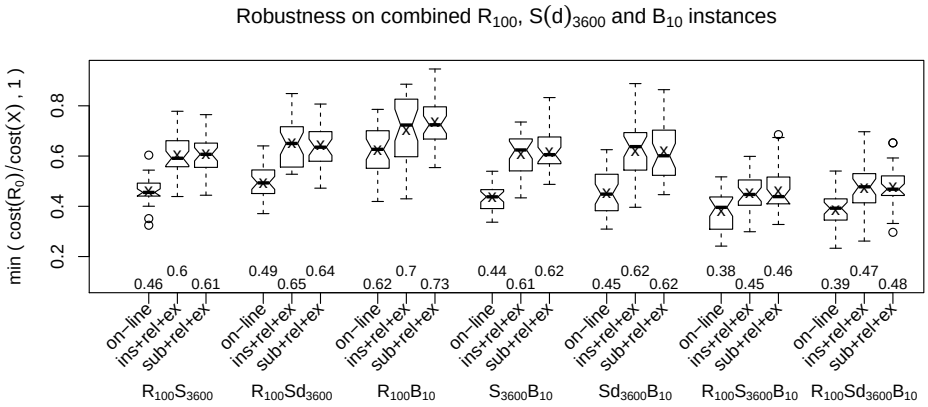


Figure 6.24: Robustness results of the three methods in different combinations of mixed-uncertainty scenarios.

significantly lower cost than the on-line optimization method. In all but the $R_{100}S_{3600}B_{10}$ scenario, this is shown by the non-overlapping notches on the boxes. For the $R_{100}S_{3600}B_{10}$ scenario, we applied a t-test to see whether the differences in the sample means are significant. For both agent methods, the test showed that there is enough evidence, with a 95% significance level ($p < 0.05$), that the sample means are smaller than the sample mean of the on-line optimization results. In particular, for the *ins+rel+ex* method the test output is: t-value $= 2.424$, the degrees-of-freedom measure is 32, the p-value $= 0.02118$, and the mean of the differences is 14126.28. The same values for *sub+rel+ex* method are: t-value $= 2.7269$, the degrees-of-freedom measure is 32, the p-value $= 0.01029$, and the mean of the differences is 19007.53.

Robustness The robustness results, as depicted in Figure 6.24 are similar to the cost results. All three methods were less robust when all three types of uncertainties were combined. In addition, the agent methods were always significantly more robust than the on-line optimization according to the notches on the box-plots.

We presented all experimental results gathered during this research. We have seen the general trend that the more uncertainties are combined, the worse the cost and robustness results become.

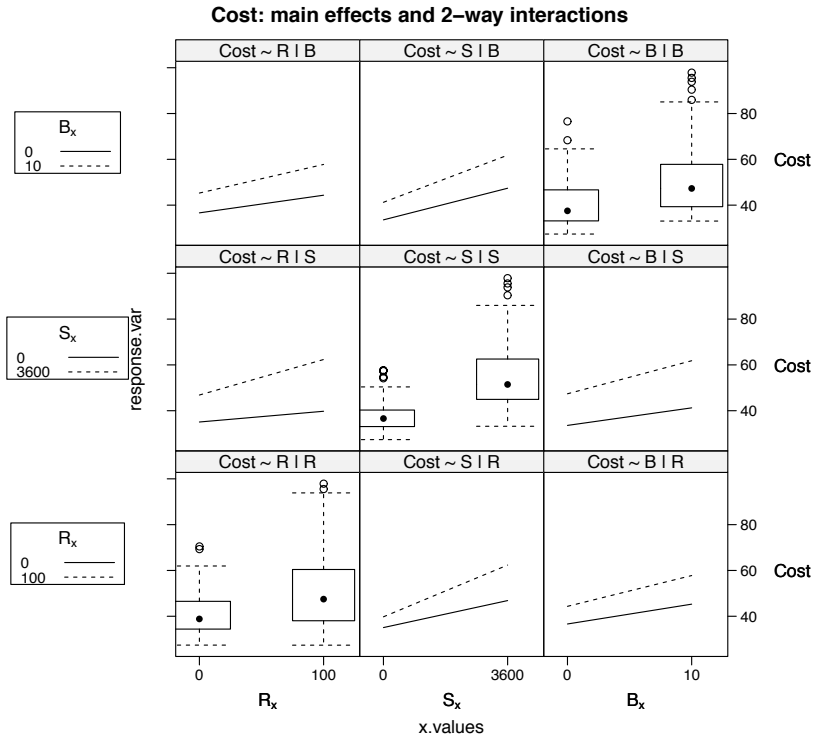


Figure 6.25: Effects of uncertainties on cost results of the sub+rel+ex *method.*

In the combined extreme scenarios of the previous sections, the agent methods always performed at least as well as the on-line optimization method, and they were always more robust than the on-line optimization. In the next section, we further analyse the influence of the uncertainty types on the methods, and we try to establish which uncertainty types have stronger effects on the methods.

The Effects of Uncertainties

After we have seen how the addition of any uncertainty type has an effect on the results, the question arises about which of the three uncertainty types has the greatest effect. Figures 6.25-6.28 display a series of plots on the main effects and 2-way interactions of the three uncertainty types *R*, *S*, and *B* (leaving *Sd* out on the basis that there was no significant difference in the results of any of the methods between the *S* and *Sd* scenarios) on the cost and robustness results of the *sub+rel+ex* and the on-line optimization methods. In all four figures, the box-plots in the diagonal, annotated as Z $\sim$ X | X, represent the main effects of the uncertainty type X on output Z. The off-diagonal plots, annotated as Z $\sim$ X | Y, show the 2-way interactions between uncertainty types X and Y, in the effect on output Z. In such plots, parallel lines reveal no interactions between the factors, while divergent of convergent lines witness the existence of interaction. The upper-left plots in the figure are mirror images of the lower-right plots.

Looking only at the diagonals of the four plots, the general trend is that service-time uncertainty with a variability of $\pm$ one hour has a greater main effect on both the cost and robustness results than 100%release-time uncertainty and the 10 truck-breakdown uncertainty. Computing the four ANOVA tables for the four figures confirms that the three main effects are significant in all four cases, the least significant being the effect of the 100% release-time uncertainty on

Table 6.8: Simple main effects of R, S, and B scenarios on the two planning methods.

	Cost			**Robustness**		
	R	*S*	*B*	*R*	*S*	*B*
on-line opt	7.36	30.48	7.92	−0.11	−0.37	−0.11
sub+rel+ex	10.10	17.19	11.05	−0.13	−0.25	−0.16

the cost results of the on-line optimization method, with an F value of 53.3152 and a p value of 3.587×10^{-12}. Table 6.8 summarizes the simple main effects of the uncertainty types, revealing that at the studied levels of uncertainties, service-time uncertainty is indeed the most influential of the three uncertainty types. Both for the cost and the robustness results, service-time uncertainty influences the planning methods more than the other uncertainty types. The truck-breakdown uncertainty with 10 trucks breaking down is the second most influential source of uncertainty, which almost always has a higher main effect than the 100% release-time scenarios (except on the robustness results of the on-line optimization method). These findings justify Hypothesis 6.4 which states that lower levels of service-time uncertainty influence the planning methods similar to higher levels of release-time or truck-breakdown uncertainties.

In the interaction plots, parallel lines represent no interaction between the two uncertainty types. The less parallel the two lines are, the higher the predicted interaction is. A high level of interaction between uncertainty types predicts that studying these uncertainties one-by-one does not give the expected results. Accordingly, in Figure 6.25 the highest interaction seems to be between the release-time and service-time uncertainties. That is, the *sub+rel+ex* agent method suffers the highest performance loss, when these two uncertainty types are present. According to the ANOVA table computed for this case, all three 2-way interactions are significant, R:S: $F = 40.0262$, $p = 1.112\times10^{-09}$; R:B: $F = 7.8376$, $p = 0.005507$; S:B: $F = 15.6636$, $p = 9.806\times10^{-05}$. However, the only 3-way interaction, R:S:B, is not significant, $F = 2.2221$, $p = 0.137279$.

For the on-line optimization method, it is the combination of release-time and truck-breakdown uncertainties that exhibits the greatest interaction effect in terms of the cost results. Even so, the main effects of service-time uncertainty is so strong that it alone is stronger than the combined effect of the other two. The strongest interaction, R:B is significant, with $F = 17.9963$, and $p = 3.096\times10^{-05}$. The weaker interaction, R:S is still significant with $F = 8.0272$, and $p = 0.004975$. The third 2-way interaction, S:B however, is not significant with $F = 0.0351$, and $p = 0.851634$. The only 3-way interaction, on the other hand, is significant in this case with $F = 7.5314$, and $p = 0.006492$.

As for the robustness results, the interaction lines of the *sub+rel+ex* agent method are mostly parallel in Figure 6.27, suggesting that there is hardly any interaction between the different sources of uncertainty. The least parallel lines are drawn in the release-time uncertainty plus service-time uncertainty plot, which also has the strongest interaction effect on the cost result of the agent method. R:S is the only interaction effect on the robustness results that is significant, $F = 10.7114$, $p = 0.001211$. All the other interaction effects are not significant: R:B: $F = 0.0449$, $p = 0.832322$; S:B: $F = 0.7217$, $p = 0.396376$; R:S:B: $F = 0.2165$, $p = 0.642129$.

Figure 6.28 suggests that there are more interaction effects in the case of the robustness

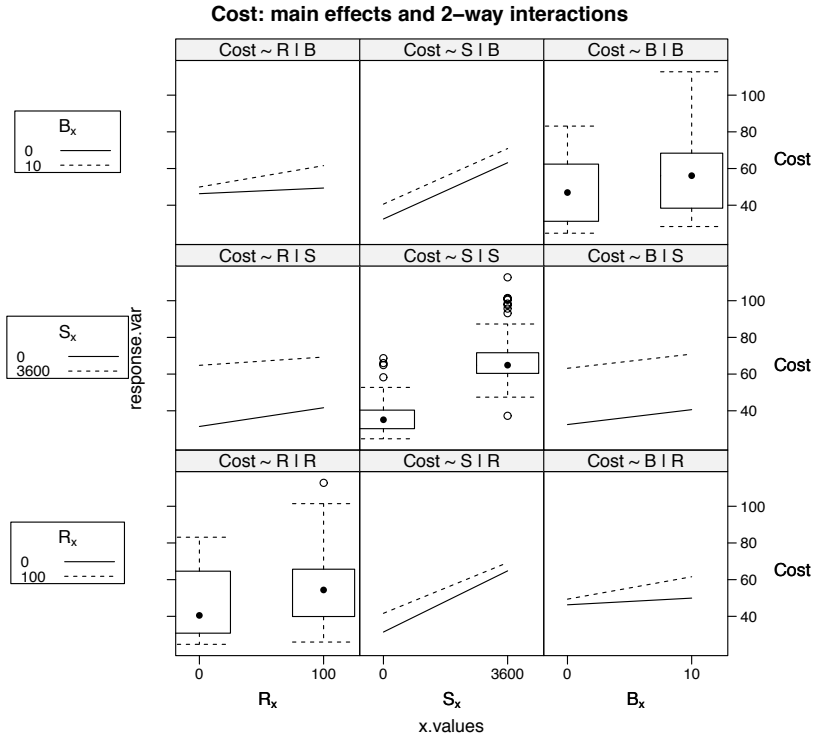


Figure 6.26: Effects of uncertainties on cost results of the on-line optimization method.

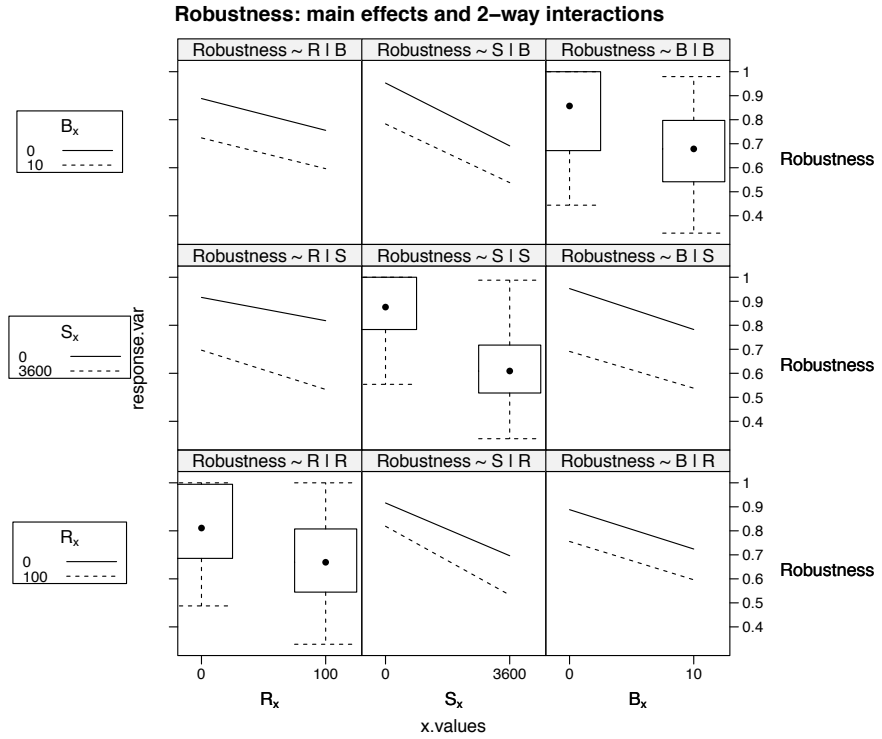


Figure 6.27: Effects of uncertainties on robustness of the sub+rel+ex *method.*

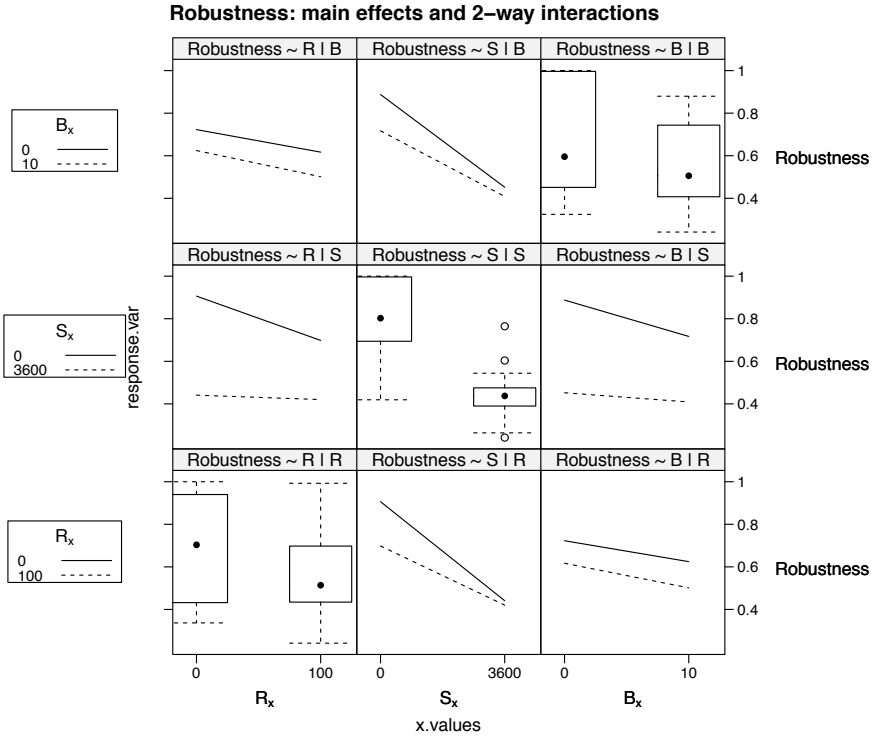


Figure 6.28: Effects of uncertainties on robustness of the on-line optimization method.

results for the on-line optimization. Only the release-time and truck-breakdown uncertainty pair seem to have no interaction effect on the robustness of the on-line optimization method. The computed ANOVA table indeed shows that the R:S interaction is significant with $F = 119.8421$, and $p < 2.2 \times 10^{-16}$, and the S:B interaction is also significant with $F = 55.4812$, and $p = 1.446 \times 10^{-12}$. In addition, the 3-way interaction is significant with $F = 8.8520$, and $p = 0.003208$, but the R:B interaction is not significant, $F = 1.0342$, $p = 0.310124$.

Summarizing the effect of uncertainties on the cost and robustness results of the agent and on-line optimization methods, we can say that each different source of uncertainty has a significant effect on the results. Interactions are generally present, a notable exception being the robustness result of the *sub+rel+ex* agent method, in which hardly any interaction is found. In addition, we note that although the effects of different uncertainties on the *ins+rel+ex* method were not discussed here, the comparison study also included this method. The effects of uncertainties on this method were found to be identical to the effects on the *sub+rel+ex* method, therefore we omitted these results for the sake of exposition.

6.4 Summary

In the beginning of this thesis on robust vehicle routing solutions, we reviewed vehicle routing problems in the operations research and agent-based logistics literature (Chapter 2). Then (in Chapter 3) we discussed different sources of uncertainty that may arise in a vehicle routing problem, and provided an extended definition of the general Transportation Problem with Uncertainties. In addition, we defined a measure of robustness to evaluate the performance of vehicle routing methods in such problems. In Chapter 4, we presented two solution approaches

to such problems, and in Chapter 5 we introduced our simulation framework to compare the different approaches. In the present chapter, to provide an answer to Research Question 2 about the effects of uncertainties on distributed and centralized planning approaches, we described the results of our computational experiments comparing the agent-based and on-line optimization approaches on the previously defined problem.

The chapter started with the formulation of a few hypotheses about the expected results. We conjectured that our substitution heuristics will improve the cost performance of the agent methods in the static instances, while having a similar performance to the agent methods without substitution in the dynamic instances (Hypothesis 6.1). Furthermore, we postulated that the on-line optimization will be better than the agent methods in static instances, but not in uncertain instances (Hypothesis 6.2). In addition, we expected agent methods to always be more robust than on-line optimization (Hypothesis 6.3). Finally, we assumed that the highest level service-time uncertainty will have the strongest effect among the defined kinds of, and at the defined levels of uncertainties (Hypothesis 6.4).

The main body of this chapter had three parts. In the first part, we described the setup of the experiments, explained how the problem instances were generated from real data, and defined the types and levels of uncertainties used in the simulations. In the second part, we provided results on the performance of different combinations of agent heuristics, concluded that Hypothesis 6.1 is true at least in R_0 and R_{100}, and selected two of the combinations to compete against the on-line optimization methods. In the third part, we presented the results of the two agent methods and the on-line optimization in the previously defined uncertainty scenarios.

First, we listed the cost and robustness results in the pure uncertainty scenarios followed by the mixed uncertainty scenarios. In the discussion of the pure-uncertainty results we concluded that, partly justifying Hypothesis 6.2, there was an interesting turning point in the cost results of the release-time and service-time uncertainty scenarios where the on-line optimization method lost its advantage to the agent methods. When the level of uncertainty was smaller than in R_{50} or S_{1800}, the on-line optimization had a smaller cost than the agent method. When the level of uncertainty was higher than in these scenarios, the agent methods dealt with the instances with lower cost.

The better performance of the on-line optimization in the static instances is based on two properties of the methods. First, the on-line method is an optimization, while the agents employ heuristics. Second, the on-line optimization uses all available information to compute the solutions, while the agent-based heuristics have restricted information available. In the static instances, the on-line optimization can capitalize on its capability of using all the available information and computing optimal plans. In the uncertainty instances however, the information available for optimization is either not much (in the release-time uncertainty cases) or not true (as in the service-time uncertainty cases). Because of this, in these instances the on-line optimization cannot benefit from considering all available information, and its performance falls behind the performance of the agent methods. This is also witnessed by the computed robustness of the methods, which, in accordance with Hypothesis 6.3, shows that the agent methods are more robust against these types of uncertainties at any tested level of uncertainty.

The pure truck-breakdown scenarios are different. Here we found no turning point, and the on-line optimization method outperformed the agent methods even in the most extreme scenario, in terms of cost. That is, the expectations of Hypothesis 6.2 were not fulfilled in these scenarios.

A possible explanation for this is that the amount of information used by the optimization but falsified by trucks breaking down, is considerably less than in the other uncertainty scenarios. That is, the on-line optimization was not misled by the breakdown instances in the same way that it was misled by the other uncertainty types. The robustness results were also different. Although the *ins+rel+ex* method was consistently more robust than the on-line optimization in every tested truck-breakdown scenario, the *sub+rel+ex* method was significantly more robust only in the scenario when 5 trucks broke down per day. The similar robustness across the methods supports the previous findings that the on-line optimization is influenced by truck breakdowns similar to the agent methods.

When we examined the results of mixed-uncertainty scenarios, we found that the agent methods were both more cost effective and more robust than the on-line optimization method in the extreme cases. Every type of uncertainty, at the extreme levels, had a significant effect on both the cost and the robustness results of all planning methods. As expected by Hypothesis 6.4, one hour service-time variations had the strongest effect, followed by the 10-truck breakdown cases, and the 100% release-time scenarios. In most of the cases, the interaction effects were also significant, except in the case of the robustness results for the agent methods. This frequent occurrence of significant interactions between the uncertainty types means that the effect of more than one source of uncertainty cannot be predicted from the results obtained under individual uncertainties.

The main conclusion that we can draw from the results is that most of our hypotheses hold, but not all and not always. The substitution algorithm improves the cost performance of the agent method in the static scenario (in R_0), but in the uncertainty scenarios the difference between the *ins+rel+ex* and *sub+rel+ex* methods was not significant. Regarding Hypothesis 6.2, there are uncertainty scenarios, namely the release-time and the service-time uncertainty scenarios, where the distributed agent methods can outperform the centralized optimization method at sufficiently high levels of uncertainty, that is Hypothesis 6.2 is true. But not all uncertainty types were found to perform in this way; in the truck-breakdown cases the agent methods could not organise cheaper transport than the on-line optimization. In terms of robustness, however, the on-line optimization was never found to be more robust than any of the agent methods, as expected by Hypothesis 6.3. Regarding our last hypothesis (Hypothesis 6.4), service-time uncertainty was found to be the most influential source of uncertainty.

Chapter 7

Conclusions and Future Work

7

This thesis discussed the application of centralized optimization-based and distributed agent-based route planning methods in vehicle routing problems with uncertainty. This problem can be approached from two dimensions. From the point of view of the planning methods, the question is which method performs better on problems with uncertainty. From the uncertainties point of view, the question is which kind of uncertainty has a larger effect on the planning methods. In this chapter, we discuss our findings from the point of view of both the planning methods, and the uncertainties. The following sections, first, answer the research questions, which is followed by a discussion of promising directions for future research.

7.1 The Comparison of Planning Methods

The first dimension that we discuss is the dimension of the planning methods. The main question raised by Research Question 2 is whether agent-based heuristics, or an on-line optimization method can handle uncertainties better. While studying this subject, we developed a new distributed heuristic algorithm for vehicle-routing problems, and compared classical heuristics, this new heuristic, and an on-line optimization algorithm on a transportation problem with uncertainties. The following sections describes our findings related to the new heuristic and the comparison of the agent-based and optimization-based approaches.

7.1.1 The Substitution Heuristics

Our new heuristic algorithm, the substitution heuristic, is described in Chapter 4, and published in (Máhr et al., 2010). The substitution heuristic is designed to solve transportation problems in a distributed manner. This new heuristic is an extension of the insertion heuristic, and is motivated by the *decommitment* idea of leveled-commitment contracts. With leveled-commitment contracts breaking a contract is allowed if the value of the new contract is greater than the commitment of the contractor to his current contract (i.e. the amount of penalty he has agreed to pay). Accordingly, if a newly inserted job fits that plan better, the new heuristic allows the substitution of an already planned job from a plan. Instead of a penalty, substitution here depends on the market position of the removed order, thereby avoiding removal of orders that would have difficulties finding another truck.

In our first exploratory experiments, described in Section 6.2, we compared several different combinations of the four agent heuristics in the static (R_0) and the maximum uncertainty release-

time (R_{100}) scenarios. The simulation results revealed that in the static case, our substitution heuristic improved the cost results more than the other two improvement heuristics (order relocation and exchange). In fact, when the substitution algorithm was applied, the improvement methods had a minimal effect. In the R_{100} case, however, any combination of the agent methods performed the same. When tested on the other uncertainty types and in the mixed scenarios, no significant difference was found between using the substitution heuristic or not. Naturally, the robustness results of the *ins+rel+ex* and *sub+rel+ex* methods were also very much alike.

The interesting conclusion is that while the substitution algorithm significantly improves the performance of the agent approach in the static case, it performs equally well as compared to the other agent method in the uncertain cases. This is unlike the on-line optimization method, which similarly outperforms the agent method without substitution (*ins+rel+ex*) in the static case, but produces worse results at the uncertain end of the scale. Therefore, we conclude that the agent method with substitution (*sub+rel+ex*) exhibits preferable behaviour than the other two in the presence of uncertainty.

7.1.2 Heuristics vs. Optimization

One of the main objectives of this research was to investigate whether a distributed computationally lightweight approach (the agent approach) could be an alternative for classical centralized optimization approaches in dynamic transportation problems. This is formulated in Research Question 2 as follows:

> *How do uncertainties in transportation problems affect centralized and distributed solution methods of such problems?*

This question is studied in this thesis to the extent of three different types of uncertainties: release-time, service-time, and truck-breakdown uncertainties. In short, our simulation results, published in (Máhr et al., 2010) and (Máhr et al., 2011), show that lightweight distributed local heuristics are competitive with centralized optimization methods, and in certain circumstances they can even outperform the centralized method.

To arrive at this conclusion, we compared the performance of three different methods using simulation. An on-line optimization method and two agent-based heuristics were compared. One of the agent methods used insertion, relocation and exchange procedures, and the other one used substitution in addition to the previous algorithms. These three methods were tested in static problem instances, instances with pure uncertainties, and instances with mixed uncertainties.

In the tests of pure-uncertainty scenarios, we found that above certain levels of uncertainty, in the cases of release-time and service-time uncertainties, the agent methods had a better cost performance than the optimization-based method. To be precise, when less than half of the orders were dynamic, the optimization performed significantly better. When more than 75% of the orders were dynamic, the agent methods performed significantly better. Between the two levels (50%-75%) the two approaches produced statistically indistinguishable ($p > 0.05$) results. When all orders were static, but the service times were varied by a maximum of 20 minutes, the on-line optimization produced better results, but when the variation was higher than that, the agent methods were more cost effective. Interestingly, in the truck-breakdown scenarios no such

tipping point was found. The on-line optimization was consistently better than the agent-methods at any (tested) level of truck breakdowns.

To see how the methods perform when more than one source of uncertainty is present, we combined the high uncertainty cases from the three uncertainty types. (We discuss the results of the independent and dependent service-time scenarios in one, as there was mostly no difference between the two cases.) Based on the conclusions of the pure-uncertainty scenarios, we combined two scenarios (R_{100}, and $S(d)_{3600}$) that favoured the agent methods, and one (B_{10}) that favoured the optimization method. It was therefore no surprise that in the combination of the release-time and service-time scenarios, the agent methods prevailed. In the notable combination of release-time and truck-breakdown uncertainties the three methods performed equally. In the combination of the service-time and truck-breakdown uncertainties the agent methods were more cost effective. These results suggest that while the maximum level of release-time and truck-breakdown uncertainties counter-balanced the effects of each other, the effect of an approximately one-hour service-time variation was stronger than the effect of 10 truck breakdowns. Based on these results, it is not surprising that in the combination of all three uncertainty types, the agent methods were significantly ($p < 0.05$) better than the on-line optimization method.

In addition to the classical cost measure, we also evaluated the three methods using our robustness measure. The summary of the results are much simpler in this case. The agent methods were found to be more robust than the on-line optimization in all (pure and combined) scenarios, except the truck-breakdown uncertainty cases. Here the *ins+rel+ex* method was not more robust than the on-line optimization, with the exception of the lowest uncertainty level (B_3), but the robustness value of the *sub+rel+ex* method practically equaled that of the on-line optimization.

7.2 The Comparison of Uncertainties

After summarizing our conclusions on the comparison of planning methods, we turn now to the different uncertainty types and on measuring their effect. The main research question that relates to this area is Research Question 1, which suggests the development of a framework for the different types of uncertainties and their effects. Additionally, Research Question 1a and 1b raise questions about the methodology for evaluating the effects of uncertainties on planning methods.

7.2.1 A Model of Transportation Problems with Uncertainties

One of the problems this thesis analyses is formulated by Research Question 1 as follows:

> *How should uncertainties in transportation problems be formalized, and what kind of framework should be developed to analyze such problems?*

As one possible answer to this question, this thesis provides an incident framework as an extension to classical transportation-problem formulations in Chapter 3. The incident model allows for the definition of different sources of uncertainty that influence plan execution independently of the underlying transportation problem. In contrast to classical transportation problems, the

uncertainty framework uses execution traces as output instead of plans, and allows the addition of different incident models. With the help of this uncertainty framework, any previously defined static problem can be turned into a dynamic transportation problem. Thanks to the separated definition of the static transportation problem and the sources of dynamism, the model facilitates the reuse of transportation problems and incident definitions. Although other formalizations are also possible, our proposed model has the interesting property that it allows for the definition of uncertainties that are arbitrary combinations of other uncertainties. In this way, complex real-world problems can be modeled by a composition of basic uncertainty types.

7.2.2 The Measure of Robustness

One of the problems in providing a comprehensive framework for uncertainties in transportation problems is formulated by Research Question 1a:

> *How can we quantify the effects of uncertainties on planning methods?*

There are multiple ways one can think about the effects of uncertainties in transportation problems. Our answer to this question is a new measure of robustness, as described in Chapter 3, that quantifies the degradation of the (cost-based) performance of a transportation-planning method under uncertainty as follows:

$$r(S) = \frac{1}{|S|} \sum_{i \in S} \min\left(1, \frac{c(\hat{\imath})}{c(i)}\right),$$

where S is a set of problem instances (a.k.a a scenario), $c(i)$ is the cost of transportation on instance i, and $\hat{\imath}$ is the static variant of i. $r(S)$ is a dimension-less, normalized measure that is based on the ratio of the method's cost-based performance in the static vs. uncertain instances. The ratio of the cost results is maximized at one, and averaged over the instances in the given uncertainty scenario. The maximization makes sure that the result is a value between 0 and 1, and it excludes disturbing effects of cases, for which a method performs better in an uncertain instance, than in its static counterpart. This, in the end, means that our proposed robustness measure quantifies the *degradation* of the cost-based results of the method due to the uncertainty present in the instances.

Our robustness measure represents one way of quantifying the effects of uncertainties on transportation planning methods. There are certainly many other measures that can be used for this purpose, and a comparison of these measures is an interesting continuation of this work.

7.2.3 Simulation Framework and Benchmarks

In order to justify our thesis that a lightweight distributed approach can be an alternative to centralized optimization in transportation problems with uncertainties, we conducted a series of computational experiments. Research Question 1b raised the issue of the proper simulation methodology:

> *What are the requirements of a simulation test-bed that is used to evaluate planning methods in transportation problems with uncertainty?*

A qualitative analysis of this problem is presented in Chapter 5, and the conclusions were published in (Máhr et al., 2011).

The problem with such experimental comparisons is that the simulation setup and the problem data can influence the results. Ideally, all published results should be based on simulations that are carried out the same way, and on data from the same problem instances. For the different planning methods, and uncertainty types studied in this thesis, however, no generally applied simulation method, or already published benchmark sets exist.

Regarding the simulation method, and the concrete answer to Research Question 1b, we concluded that a simulation framework of transportation problems with uncertainties should meet the following three requirements to provide a fair comparison of on-line planning approaches. It should be able to simulate unforeseen events; it should separate the planning algorithm and the simulation of the execution of the plans; and it should simulate the execution of the plans in real time. The first requirement is obvious, and the other two requirements aim to ensure that different kinds of planning methods can be compared, and that the computation time of these methods are considered by the framework.

Regarding the data for the problem instances, since benchmark data for transportation problems is rare, we based our comparison study on instances generated from a real dataset. We were fortunate to have access to the operational data of a mid-sized Dutch transportation company. We generated several different problem instances that reflect the every-day uncertainties experienced by the planners of the company. The prepared instances contained release-time, two types of service-time, and truck-breakdown uncertainty cases, and combinations of these uncertainties. To help subsequent studies on dynamic transportation methods, we published the generated release-time uncertainty instances (Srour et al., 2010), and contributed them to MIPLIB[1], an electronically available library of both pure and mixed integer programs.

7.3 Future Work

When experimenting with computer algorithms, there are plenty of choices one has to make to constrain the number of different paths to explore. For every project, there is a limited amount of time, and with the approach of a deadline the list of ideas in the drawer labeled 'Future Work' grows. In this section, we review certain aspects of nearly all components of the evaluation framework, and discuss alternatives to the choices we made. First, we discuss the simulated sources of uncertainty, then the planning methods, and finally the simulator.

7.3.1 On the sources of Uncertainty

In this thesis we considered three different sources of uncertainty: release-time, service-time, and truck-breakdown uncertainty. When we described our simulation environment in Chapter 5, we listed two additional uncertainty types which can be simulated in this framework, but which were omitted for now. These were the travel-time uncertainty and the communication uncertainty types. Although we did not simulate such uncertainties in this thesis, they are nevertheless interesting candidates for future research.

[1] http://miplib.zib.de/

Regarding the levels of uncertainties, it is still unclear what is the best way of comparing two sources of uncertainty in general. In this thesis we compared the given levels of uncertainties based on the cost and robustness results. Another approach to this would be to ask, given two different sources of uncertainty, what are the respective levels of uncertainty that produce the same results using a given planning method? This question was not analyzed in this thesis, but it would be a natural continuation of the present work.

In addition to the aforementioned uncertainty types, there are several more sources of uncertainty that can be considered in transportation problems. One example is the question of time windows. We considered a fixed narrow (two-hour) time window at the customer location, and wide non-restrictive time windows at the sea terminals. In our case, this choice was motivated by the practical experience of the LSP suggesting that sea terminals tend to be forgiving with respect to the times trucks visit them, but customers are less so. Nevertheless, it could be interesting to compare the planning methods in scenarios with different time-window sizes, or with non-uniform (possibly bursty) distributions.

Another fixed setting related to our uncertainty scenarios is the type of distribution used to generate the instances. Since time windows are distributed uniformly, dynamic orders in release-time uncertainty scenarios arrive uniformly throughout the day. The variation of service times, in service-time uncertainty scenarios, is generated according to a uniform distribution between given bounds. Also, in truck-breakdown scenarios, trucks break down uniformly throughout the day. It would be interesting to experiment with similar scenarios, but with more 'bursty' distributions (some type of beta distribution, perhaps).

In addition to the different types of uncertainty, the other main theme in this thesis is the comparison of agent-based and optimization-based planning methods. The next section describes a few ideas for future research regarding these.

7.3.2 On Planning Methods

By adapting and developing the two approaches to vehicle routing, our goal was to provide a distributed and a centralized transportation planning mechanism that can generate state-of-the-art solutions to truckload vehicle routing problems. In the following we identify key points in our methods that provide options for further research, and we formulate open questions that address these possibilities. We proceed from concrete to more general questions. We start with a specific issue in the insertion-substitution algorithm, then we discuss the possibility of considering competitive agents, and finally we mention the option of combining these approaches and stochastic vehicle-routing approaches.

The insertion-substitution algorithm described in Section 4.1.1 allows truck agents to reconsider earlier decisions, and forgo existing contracts by releasing orders agents. Released order agents re-auction themselves and make new contracts that might cause other order agents to be released by their trucks. To prevent an infinite loop of substitution, a simple rule is applied in the algorithm that allows a substitution to happen only if it yields a more efficient plan (with less empty travel). Although this rule prevents infinite substitution chains, it is too strict in some sense. Only those substitution chains for which the overall empty travel time decreases at every substitution are allowed. It is possible, however, that certain highly efficient plans are only reachable through substitutions that increase empty travel time. Therefore, one could allow

substitutions that temporarily decrease quality, in the hope that a better quality solution is found. A possible approach would be to fix a maximum number of cost-increasing substitutions along a chain. This would allow some steps to go uphill, but in the end the chain would ultimately end. An alternative to this approach is to fix an upper limit on the allowed increase of empty travel time. This is a more flexible approach as it would permit use of the allowed increase in a few big steps, or in several small steps. Furthermore, well-known meta-heuristics can also be applied here. In a simulated annealing approach one would allow a cost-increasing step with a certain probability, which would cool off by time. A tabu-search approach can also be employed to remember earlier substitutions in the chain as a means to prevent looping. All these approaches might possibly produce more efficient plans in the distributed solution method. The interesting question is whether such enhanced versions of the algorithm retain the robustness properties of the current distributed method, or whether they would be closer to the on-line optimization method?

Moving on to the question of competition in the agent system, we first have to note that the multi-agent system described in previous sections is a cooperative-agent system. Agents accept solutions by which they are worse off individually, as long as other agents can benefit from it. In some situations, however, a cooperative system is not feasible. Typically, when agents represent different, competing companies, one cannot assume cooperation among the agents. Additionally, in such cases, privacy issues become very important, meaning that the agents will not share sensitive information with each other. Some of the techniques applied in our solution are still applicable in a competitive setting, while others raise issues. The applied auction mechanism for example, would induce strategic bidding on behalf of the truck agents. In a sequence of auctions, in contrast to our approach, bidders would have an incentive to speculate on future items, and bid a value different than their true value for an item. Another problematic point is the exchange method by which truck agents share sensitive planning information with each other and consider group utility when deciding on an exchange. This is not acceptable in a competitive system. There are further issues with the substitution technique too. One cannot break contracts in a competitive environment without offering a penalty. It is not clear how penalties should be set in a system that includes strategic agents. Clearly, some of the agent algorithms used in this thesis cannot be applied, if a competitive agent system is required. Although, it seems to be a difficult direction, it would be interesting to see how a group of truly competitive agent would perform compared to our cooperative ones.

Regarding the optimization-based approach, we should point out that its performance depends on the performance of the MIP solver on which it is based. The results discussed in this thesis are generated using the SCIP[2] solver, which claims to be the fastest non-commercial mixed-integer programming solver. We also, however, ran a subset of the experiments using CPLEX[3] as the MIP solver. We found the performance of the on-line optimization approach better (cost-wise) with the CPLEX solver (with results published in (Máhr et al., 2010)) than with the SCIP solver (with results published here). The difference can be attributed mainly to the speed of the solvers. CPLEX was faster in computing solutions, therefore it rejected orders due to the lack of a feasible solution less frequently than SCIP. Additionally, CPLEX handled numerical instability issues much better than the SCIP solver. Even though the actual performance of the on-line optimization

[2] http://scip.zib.de/

[3] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

method can be improved by using a better MIP solver, the trends found using the different solvers are the same. Additionally, the same arguments naturally hold for the agent-based methods too. Improving the algorithms used by the agents could improve their performance.

Finally, we mention stochastic approaches. Both approaches described in Chapter 4 are deterministic reactive planning methods, which means that they make plans ignoring the possibilities of unforeseen events, and they adjust the plans after the events occur. Another popular way of dealing with uncertainty is to assume probability distributions over the possible outcomes of the events, and produce plans that are optimal in the expectation of these probabilities. A possible question is whether such stochastic approaches combined with reactive approaches could produce more robust executions than any of the two approaches separately.

7.3.3 On the Simulation of Trucks

In our simulation environment, route planning and the simulation of truck movements are separated into separate components. This raises the issue of time synchronization. To stay in concert, all components need to synchronise their clocks with the simulator's clock. Furthermore, the simulator is required to be a real-time simulator able of explicitly considering computation times. Many agent-based discrete-time simulators employ a turn-based simulation technique, thereby hiding the effects of computation-time differences. This is avoided by a real-time simulation that has the (required) consequence that the planners have only a limited time to calculate their plans before the world changes. Nevertheless, spending a month on simulating the execution of a single month is, of course, unacceptable. As a workaround to these problems, the simulator has been made capable of simulating a given factor faster or slower than real time, and the planning components have been made to synchronize to the simulator. The practical problem with this setup is to find out how fast the simulator should run. After some exploratory experimentation, we found that 6 times the real time is a good compromise, which allows enough time for the planners to come up with plans. An alternative solution would have been to change the speed of simulation while running. The simulator could slow down at times of change to allow the planners to react, while it could speed up at times when nothing happens to rush through the uninteresting parts of the simulation. This, however, would require more careful and prompt time synchronisation, and would complicate the operations of both the planning components and the planners.

In the above sections, we summarized possible alternative settings of the planning algorithms, the simulator, and the sources of uncertainty. We believe that the open issues mentioned here are interesting directions of future research, and hope that we will have the chance to follow some of these directions, and that others will also find them worthy of further elaboration.

Bibliography

Aknine, S., Pinson, S., and Shakun, M. F. (2004). An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, 8(1):5–45. ISSN 1387-2532.

Altinkemer, K. and Gavish, B. (1991). Parallel savings based heuristic for the delivery problem. *Operations Research*, 39:456–469.

Alvarenga, G., de Abreu Silva, R., and Mateus, G. (2005). A hybrid approach for the dynamic vehicle routing problem with time windows. In *Hybrid Intelligent Systems, 2005. Fifth International Conference on*, page 7pp.

Andersson, M. and Sandholm, T. W. (1998). Leveled commitment contracts with myopic and strategic agents. In *Proceedings of Fifteenth National Conference on Artificial Intelligence*, pages 38–44.

Bachem, A., Hochstättler, W., and Malich, M. (1996). The simulated trading heuristic for solving vehicle routing problems. *Discrete Applied Mathematics*, 65(1-3):47–72.

Bataineh, S. M. (2005). Toward an analytical solution to task allocation, processor assignment, and performance evaluation of network processors. *Journal of Parallel and Distributed Computing*, 65(1):29–47.

Beamon, B. (1998). Supply chain design and analysis: Models and methods. *International Journal of Production Economics*, 55(3):281–294.

Beamon, B. (1999). Measuring supply chain performance. *International Journal of Operations and Production Management*, 19(3):275–292.

Bell, M. G. H., Trozzi, V., Hosseinloo, S. H., Gentile, G., and Fonzone, A. (2010). Time-dependent hyperstar algorithm for robust vehicle navigation in time-dependent stochastic road networks.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, dover paperback edition (2003) edition. ISBN 0486428095.

Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization. *Math. Oper. Res.*, 23:769–805. ISSN 0364-765X.

Bertsimas, D. J. and Simchi-Levi, D. (1996). A new generation of vehicle routing research: Robust algorithms, addressing uncertainty. *OPERATIONS RESEARCH*, 44(2):286–304.

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

Bianchi, L. (2000). Notes on dynamic vehicle routing - the state of the art -. Technical Report IDSIA-05-01, ISDIA, Switzerland.

Brandt, F., Brauer, W., and Weiss, G. (2000). Task assignment in multiagent systems based on vickrey-type auctioning and leveled commitment contracting. In *Cooperative Information Agents*, pages 95–106.

Brandt, F. and Weiss, G. (1999). Exploring auction-based leveled-commitment contracting - part i: English-type auctioning. Technical Report FKI-23499, Institut für Informatik, Technische Universität München.

Brandt, F. and Weiss, G. (2000). Exploring auction-based leveled-commitment contracting - part ii: Dutch-type auctioning. Technical Report FKI-23499, Institut für Informatik, Technische Universität München.

Breedam, A. V. (1994). *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selection of Problems with Vehicle-Related, Customer-Related, and Time-Related Constraints*. Ph.D. thesis, University of Antwerp.

Bullnheimer, B., Hartl, R. F., and Strauss, C. (1997). Applying the ant system to the vehicle routing problem. In *2nd International Conference on Metaheuristics*. Sophia-Antipolis, France.

Bürckert, H.-J., Fischer, K., and Vierke, G. (2000). Holonic transport scheduling with teletruck. *Applied Artificial Intelligence*, 14(7):697–725.

Campbell, A. and Savelsbergh, M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science*, 38(3):369–378.

Chen, Z.-L. and Xu, H. (2006). Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows. *Transportation Science*, 40(1):74–88.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Maudet, N., and guez-Aguilar, J. A. R. (2005). Multiagent resource allocation. *The Knowledge Engineering Review*, 20(2):143–149. ISSN 0269-8889.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., tre, M. L., Maudet, N., Padget, J., Phelps, S., guez-Aguilar, J. A. R., and Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica*, 30:3–31.

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. (2001a). VRP with time windows. pages 157–193.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001b). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936.

Dantzig, G. B. and Ramser, R. (1959). The truck dispatching problem. *Management Science*, 6:80–91.

Davidsson, P., Holmgren, J., Kyhlbäck, H., Mengistu, D., and Persson, M. (2007a). Applications of agent based simulation. In Antunes, L. and Takadama, K., editors, *Multi-Agent-Based Simulation VII*, volume 4442 of *Lecture Notes in Computer Science*, pages 15–27. Springer Berlin / Heidelberg. 10.1007/978-3-540-76539-4_2.

Davidsson, P., Holmgren, J., Persson, J. A., and Ramstedt, L. (2008). Multi agent based simulation of transport chains. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, AAMAS '08, pages 1153–1160. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. ISBN 978-0-9817381-1-6.

Davidsson, P., Persson, J. A., and Holmgren, J. (2007b). On the integration of agent-based and mathematical optimization techniques. In *Proceedings of the 1st KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, KES-AMSTA '07, pages 1–10. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-72829-0.

Desrochers, M. and Verhoog, T. (1989). A matching based savings algorithm for the vehicle routing problem. Cahiers du GERAD G-89-04, École des Hautes Études Commerciales de Montréal, Canada.

Desrosiers, J., Laporte, G., Sauve, M., Soumis, F., and Taillefer, S. (1988). Vehicle routing with full loads. *Computers and Operations Research*, pages 219–226.

Desrosiers, J., Soumis, F., and Desrochers, M. (1984). Routing with time windows by column generation. *Networks*, 14(4):545–565.

Djadane, M., Goncalves, G., Hsu, T., and Dupas, R. (2006). Dynamic vehicle routing problems under flexible time windows and fuzzy travel times. In *Service Systems and Service Management, 2006 International Conference on*, volume 2, pages 1519–1524.

Dorer, K. and Calisti, M. (2005). An adaptive solution to dynamic transport optimization. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 45–51. ACM Press, New York, NY, USA. ISBN 1-59593-093-0.

Dueck, G. (1993). New optimization heuristics :the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86–92.

Dueck, G. and Scheurer, T. (1990). Threshold accepting: A general purpose optimization algorithm. *Journal of Computational Physics*, 90:161–175.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operations Research*, 54:7–22.

Dutta, P. S., Jennings, N. R., and Moreau, L. (2006). Adaptive distributed resource allocation and diagnostics using cooperative information-sharing strategies. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 826–833. ACM Press, New York, NY, USA. ISBN 1-59593-303-4.

Erera, A. L., Morales, J. C., and Savelsbergh, M. W. P. (2010). The vehicle routing problem with stochastic demand and duration constraints. *Transportation Science*, 44(4):474–492.

European Commission, Directorate-General for Energy and Transport and Eurostat (2007). Energy and transport in figures. Technical report, European Union. Part 3 : Transport.

Feuerstein, E. and Stougie, L. (2001). On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105. ISSN 0304-3975.

Figliozzi, M. A., Figliozzi, M. A., and Figliozzi, M. A. (2007). Pricing in dynamic vehicle routing problems. *Transportation Science*, 41:302–318.

Fischer, K., Muller, J. P., Pischel, M., and Schier, D. (1995). A model for cooperative transportation scheduling. In *Proceedings of the First International Conference on Multiagent Systems.*, pages 109–116. AAAI Press / MIT Press, Menlo park, California.

Fisher, M. L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124.

Fisher, R. C. (2000). "Numbers are Essential": Victory in the North Atlantic Reconsidered, March-May 1943. `http://www.familyheritage.ca/Articles/victory1943.html`.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. d., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106:491–511. ISSN 0025-5610. 10.1007/s10107-005-0644-x.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA. ISBN 0716710447.

Gendreau, M. and Potvin, J.-Y. (1998). *Fleet management and logistics*, chapter Dynamic vehicle routing and dispatching, pages 115–126. Kluwer, Dordrecht.

Gerkey, B. and Mataric, M. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939.

Gillet, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22:340–349. Sweep heuristic.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549. Tabu search.

Haghani, A. and Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers and Operations Research*, 32(11):2959–2986. ISSN 0305-0548.

Hirsh, A. E. and Gordon, D. M. (2001). Distributed problem solving in social insects. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):199–221. ISSN 1012-2443.

Hoen, P. J. and Poutré, J. A. L. (2003). A decommitment strategy in a competitive multi-agent transportation setting. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1010–1011. ACM Press, New York, NY, USA.

Holmgren, J., Persson, J., and Davidsson, P. (2009). Agent-based dantzig-wolfe decomposition. In Håkansson, A., Nguyen, N., Hartung, R., Howlett, R., and Jain, L., editors, *Agent and Multi-Agent Systems: Technologies and Applications*, volume 5559 of *Lecture Notes in Computer Science*, pages 754–763. Springer Berlin / Heidelberg. 10.1007/978-3-642-01665-3_76.

Housroum, H., Hsu, T., Dupas, R., and Goncalves, G. (2006). A hybrid ga approach for solving the dynamic vehicle routing problem with time windows. In *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, volume 1, pages 787–792.

Ichouaa, S., Gendreaua, M., and Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396.

Jaillet, P. (1985). *Probabilistic Travelling Salesman Problems*. Ph.D. thesis, Department of Civil Engineering, MIT.

Jaillet, P. (1991). Probabilistic routing problems in the plane. *Operational Research 90*, 29(10):675–688.

Jennings, N. R. (1996). Coordination techniques for distributed artificial intelligence. In O'Hare, G. M. P. and Jennings, N. R., editors, *Foundations of Distributed Artificial Intelligence*, pages 187–210. John Wiley & Sons.

Jih, W.-R. and Yung-Jen Hsu, J. (1999). Dynamic vehicle routing using hybrid genetic algorithms. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 453–458vol.1.

Jung, S. and Haghani, A. (2001). A genetic algorithm for the time dependent vehicle routing problem, transportation research record. *Journal of Transportation Research Board*, 1771:161–171.

Kelly, J. and Xu, J. P. (1996). A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393.

Kinderwater, G. A. P. and Savelsbergh, M. W. P. (1997). *Local Search in Combinatorial Optimization*, chapter Vehicle Routing: Handling Edge Exchanges, page 337–360. Wiley, Chichester, UK.

Klein, R. (1992). Walking an unknown street with bounded detour. *Comput. Geom. Theory Appl.*, 1:325–351. ISSN 0925-7721.

Kohout, R. and Erol, K. (1999). In-time agent-based vehicle routing with a stochastic improvement heuristic. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 864–869. American Association for Artificial Intelligence, Menlo Park, CA, USA. ISBN 0-262-51106-1.

Krauth, E., Moonen, H., Popova, V., and Schut, M. (2005). Performance measurement and control in logistics service providing. *The ICFAIAN Journal of Management Research*, 4(7).

KulKarni, R. and Bhave, P. (1985). Integer programming formulations of vehicle routing probelms. *European Journal of Operations Research*, 20:58–67.

Kuorilehto, M., Hännikäinen, M., and Hämäläinen, T. D. (2005). A survey of application distribution in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 5(5):774–788. ISSN 1687-1472.

Lambrechts, O., Demeulemeester, E., and Herroelen, W. (2010). Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals of Operations Research*, pages 1–22. ISSN 0254-5330. 10.1007/s10479-010-0777-z.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358.

Larsen, A., Madsen, O., and Solomon, M. (2002). Partially dynamic vehicle routing - models and algorithms. *Journal of the Operational Research Society*, 53(6):637–646.

Larson, K. and Sandholm, T. (2004). Designing auctions for deliberative agents. In *AAMAS-04 workshop on Agent Mediated Electronic Commerce (AMEC-04)*, pages 225–238. New York.

Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., and Shmoys, D. B., editors (1985). *Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Interscience Series in Discrete Mathematics. Wiley, New York.

Lenstra, J. K. and Kan, A. H. G. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Leong, H. W. and Liu, M. (2006). A multi-agent algorithm for vehicle routing problem with time window. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 106–111. ACM Press, New York, NY, USA. ISBN 1-59593-108-2.

Lerman, K., Jones, C., Galstyan, A., and Matarić, M. J. (2006). Analysis of dynamic task allocation in multi-robot systems. *Int. J. Rob. Res.*, 25(3):225–241. ISSN 0278-3649.

Likhodedov, A. and Sandholm, T. (2003). Auction mechanism for optimally trading off revenue and efficiency. In *EC '03: Proceedings of the 4th ACM conference on Electronic commerce*, pages 212–213. ACM Press, New York, NY, USA. ISBN 1-58113-679-X.

Mahmassani, H. S., Kim, Y.-J., and Jaillet, P. (2000). Local optimization approaches to solve dynamic commercial fleet management problems. *Transportation Research Record*, 1733:71–79.

Máhr, T. and de Weerdt, M. (2005). Distributed agent platform for advanced logistics. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 155–156. ACM Press, New York, NY, USA. ISBN 1-59593-093-0. Accepted as poster.

Máhr, T. and de Weerdt, M. (2007). Auctions with arbitrary deals. In Mařík, V., Vyatkin, V., and Colombo, A., editors, *HoloMAS 2007*, volume 4659 of *LNAI*, pages 37–46. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-74478-8.

Máhr, T. and de Weerdt, M. M. (2004). Distributed agent platform for advanced logistics. In *Proceedings of the Belgium-Dutch Conference on Artificial Intelligence (BNAIC-04)*, pages 395–396. BNVKI.

Máhr, T. and de Weerdt, M. M. (2006). Multi-attribute vickrey auctions when utility functions are unknown. In Schobbens, P.-Y., Vanhoof, W., and Schwanen, G., editors, *Proceedings of the Belgium-Dutch Conference on Artificial Intelligence (BNAIC-06)*, pages 221–227. BNVKI. ISBN 1568-7805.

Máhr, T., de Weerdt, M. M., Srour, J., and Zuidwijk, R. (2006). Incorporating expert knowledge in decision support for logistics: Balancing costs and customer relations. In *Proceedings of the 9th TRAIL congress: TRAIL in Motion*, pages 217–232. TRAIL Research School.

Máhr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2008). Agent performance in vehicle routing when the only thing certain is uncertainty. In *Proceedings of the workshop on Agents in Traffic and Transportation (ATT)*.

Máhr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2010). Can agents measure up? a comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research Part C: Emerging Technologies*, 18(1):99 – 119. ISSN 0968-090X. Information/Communication Technologies and Travel Behaviour; Agents in Traffic and Transportation.

Máhr, T., Srour, J., and de Weerdt, M. M. (2011). Using simulation to evaluate how multi-agent transportation planners cope with truck breakdowns. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control*. IEEE.

Máhr, T., Valk, J., and van Tooren, P. (2005). A logistical multi-agent system. In *Proceedings of International Conference on Self-Organization and Adaptation of Multi-agent and Grid Systems: Emerging Works on Autonomic Informatics*. SOAS'2005.

Malandraki, C. and Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulation, properties and heuristic algorithms. *Transportation Science*, 26:185–200.

Mes, M., van der Heijden, M., and van Harten, A. (2007). Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1):59–75.

Mitrovic-Minic, S. (1998). Pickup and delivery problem with time windows: A survey. TR 1998-12, Simon Fraser University, Burnaby, BC, Canada.

Perugini, D., Lambert, D., Sterling, L., and Pearce, A. (2003a). A distributed agent approach to global transportation scheduling. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 18–24.

Perugini, D., Wark, S., Zschorn, A., Lambert, D., Sterling, L., and Pearce, A. (2003b). Agents in logistics planning – experiences with the coalition agents experiment project. In *Agents at work: Deployed applications of autonomous agents and multi-agent systems, a workshop at the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*. Melbourne, Australia.

Plum, D. and Ali, H. H. (2005). An evolutionary approach to vehicle routing problem with dynamic time and precedence relationships. *Journal of Computational Methods in Science and Engineering*, 5(1):57–66.

Poladian, V., Sousa, J. P., Garlan, D., and Shaw, M. (2004). Dynamic configuration of resource-aware services. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 604–613. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-2163-0.

Powell, W., Jaillet, P., and Odoni, A. (1995). *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter Stochastic and Dynamic Networks and Routing. North-Holland, Amsterdam.

Psaraftis, H. (1980). A dynamic programming solution to single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2).

Psaraftis, H. (1983). An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17(3).

Psaraftis, H. (1988). *Vehicle Routing: Methods and Studies*, chapter Dynamic vehicle routing problem, pages 223–248. Elsevier Science Publishers, Amsterdam.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *Annals of Operations Research*, 61(1):143–164.

Rochat, Y. and Taillard, E. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167.

Rosenschein, J. S. and Zlotkin, G. (1994). *Rules of encounter: designing conventions for automated negotiation among computers*. MIT Press, Cambridge, MA, USA. ISBN 0-262-18159-2.

Ryan, D. M., Hjorring, C., and Glover, F. (1993). Extensions of the petal method for vehicle routing. *Journal of the Operational Research Society*, 44:289–296. Petal algorithm.

Sagar, G., Sarje, A. K., and Ahmed, K. U. (1989). Task allocation techniques for distributed computing systems: a review. *Journal of Microcomputer Applications*, 12(2):97–105. ISSN 0745-7138.

Sandholm, T., Sikka, S., and Norden, S. (1999). Algorithms for optimizing leveled commitment contracts. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 535–541. Stockholm, Sweden.

Sandholm, T. W. (1995). Limitations of the Vickrey auction in computational multiagent systems. In Lesser, V., editor, *Proceedings of the First International Conference on Multi–Agent Systems*. MIT Press.

Sandholm, T. W. (1999). Distributed rational decision making. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. The MIT Press, Cambridge, MA, USA.

Sandholm, T. W. and Lesser, V. R. (2001). Leveled commitment contracts and strategic breach. *Games and Economic Behaviour*, 35:212–270.

Savelsbergh, M. and Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, 29(1):17–29.

Schillo, M., Kray, C., and Fischer, K. (2002). The eager bidder problem: a fundamental problem of dai and selected solutions. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 599–606. ACM Press, New York, NY, USA. ISBN 1-58113-480-0.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J.-F., editors, *Fourth International Conference on Principles and Practice of Constraint Programming (CP '98)*, pages 417–431. Springer-Verlag.

Shepherd, C. and Günter, H. (2006). Measuring supply chain performance: current research and future directions. *International Journal of Productivity and Performance Management*, 55(3/4):242–258.

Sleator, D. D. and Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Commun. ACM*, 28:202–208. ISSN 0001-0782.

Smith, R. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *Computers, IEEE Transactions on*, C-29(12):1104–1113.

Smith, R. (1981). Correction to "the contract net protocol: High-level communication and control in a distributed problem solver". *Computers, IEEE Transactions on*, C-30(5):372–372.

Sodhi, M. S. (2007). What about the 'O' in O.R.? *OR/MS Today*, page 12.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

Sommer, C. and Dressler, F. (2008). Progressing toward realistic mobility models in VANET simulations. *Communications Magazine, IEEE*, 46(11):132 –137. ISSN 0163-6804.

Srour, F. J., Máhr, T., de Weerdt, M. M., and Zuidwijk, R. A. (2010). MIPLIB truckload PDPTW instances derived from a real-world drayage case. In *ERIM report series research in management Erasmus Research Institute of Management*. Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam (ERIM is the joint researchinstitute of the Rotterdam School of Management, Erasmus University and the Erasmus School of Economics (ESE) at Erasmus University Rotterdam).

Srour, J., Máhr, T., de Weerdt, M. M., Zuidwijk, R., and Moonen, H. (2008). *Smart Business Networks: A new business paradigm*, chapter Performance Evaluation within a Networked Enterprise: Balancing Local Objectives and Network Relations, pages 279–304. ERIM Electronic series. SBNi. ISBN 978-90-5892-159-8. See `http://www.erim.eur.nl/ERIM/Research/Centres/SBNi/SBNi_Updates/News%20Detail?p_item_id=5120098&p_pg_id=93`.

Sungur, I. (2009). *The Robust Vehicle Routing Problem*. VDM Verlag. ISBN 978-3639125009.

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.

Sycara, K., Roth, S., Sadeh, N., and Fox, M. (1991). Resource allocation in distributed factory scheduling. *Expert, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 6(1):29–40.

Taillard, E. D., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31:170–186.

Thompson, P. and Psaraftis, H. (1993). Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations research*, 41(5).

Tian, Y., Song, J., Yao, D., and Hu, J. (2003). Dynamic vehicle routing problem using hybrid ant system. In *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, volume 2, pages 970–974vol.2.

Toth, P. and Vigo, D. (2002a). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1-3):487–512. ISSN 0166-218X.

Toth, P. and Vigo, D., editors (2002b). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS J. on Computing*, 15(4):333–346. ISSN 1526-5528.

van der Putten, S., Robu, V., Poutré, H. L., Jorritsma, A., and Gal, M. (2006). Automating supply chain negotiations using autonomous agents: a case study in transportation logistics. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1506–1513. ACM Press, New York, NY, USA. ISBN 1-59593-303-4.

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37.

Woensel, T. V., Kerbacheb, L., Peremansc, H., and Vandaele, N. (2008). Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research*, 186:990–1007.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester, England.

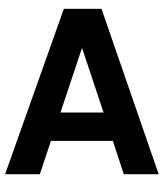
Xu, H., Long Chen, Z., Rajagopal, S., Arunapuram, S., and Inc, M. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364.

Yang, J., Jaillet, P., and Mahmassani, H. (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38(2):135–148. ISSN 1526-5447.

Yang, J., Jaillet, P., and Mahmassani, H. S. (1999). On-line algorithms for truck fleet assignment and scheduling under real-time information. *Transportation Research Record*, 1667:107–113.

Appendix A

Scalability



In the scenarios discussed in previous sections, we varied the level of uncertainty from low to high, but the size of the problem was kept fixed. All instances contained 65 containers and 40 trucks. This was motivated by the real-world case of the LSP from which we received the data, but it was also ideal for the planning methods. This problem size was easily handled by both the on-line optimization and the agent methods.

Recall that the on-line optimization method builds and solves a mixed-integer problem every 30 seconds (except the first iteration). The branch-and-bound algorithm used to solve the problem has exponential time and space requirements, meaning that this method quickly becomes computationally intractable as the problem size grows.

In contrast, the agent methods consist of polynomial algorithms that scale much better than the on-line optimization method. In Figure A.1, we depict the change of the cost results of the three methods in static scenarios with an increasing number of orders. The three methods scale very similarly up to the *90 orders* scenario, beyond which the on-line optimization runs out of memory, and cannot solve the problem any more. We should note, that neither of the two methods (agent-based or on-line optimization) had an issue with computing the solutions in time. Even though the simulations ran 6 times faster than real time, both the on-line optimization and the agent methods could react to changes in time. The problem introduced by larger instances was a function of memory: the MIP no longer fit into the available 2GB memory.

Even though the agents can handle larger problem instances, due to the distributed nature of the agent model, the question of communication complexity arises. As shown at the end of Section 4.1.1, the messaging complexity of the agent algorithms is polynomial (with the exception of the substitution heuristic) in a similar way to the computational complexity.

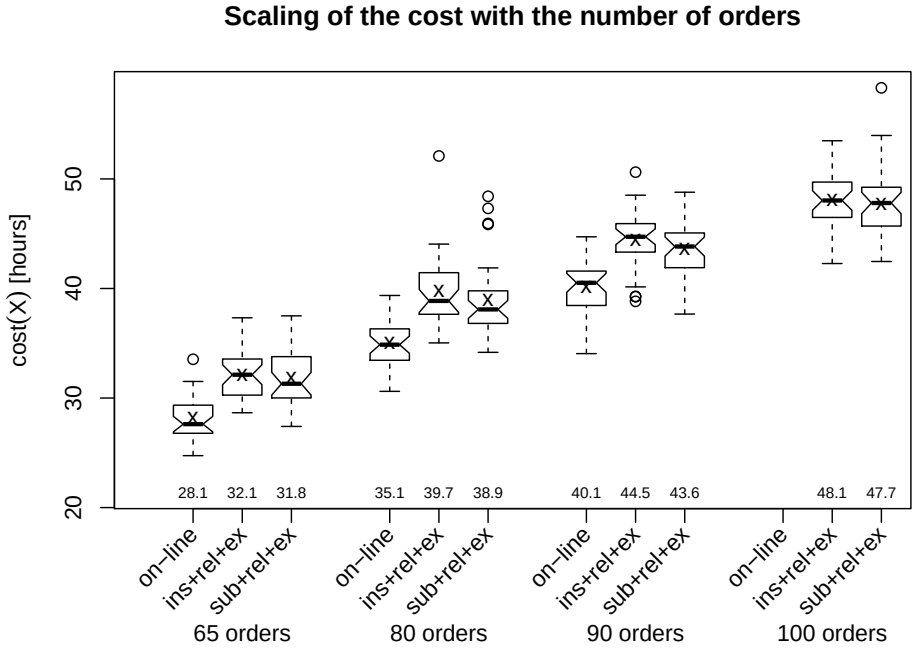


Figure A.1: The scaling of the cost results with the number of orders.

Appendix B

Normal Distribution in Results

B

During the analysis of the results, in some cases, we performed t-tests to assess the difference between the results of two methods. These t-tests rely on that the sample data has normal distribution. The following figures present normal-quantile plots, in which the close linearity of the data points suggests a normal distribution in the data.

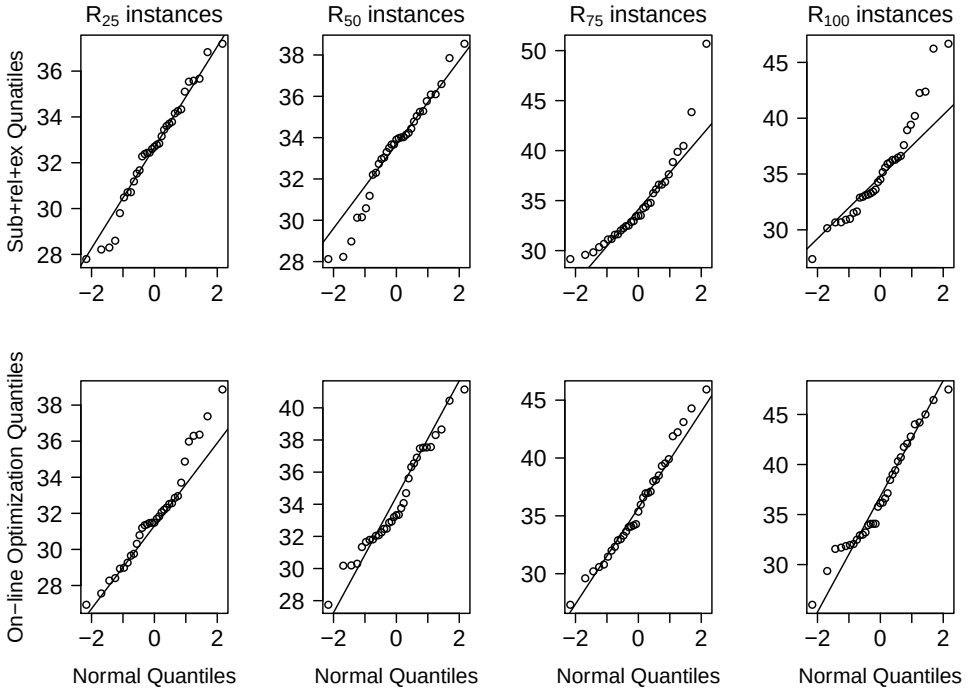


Figure B.1: Normal quantile plots of cost results on R_* scenarios.

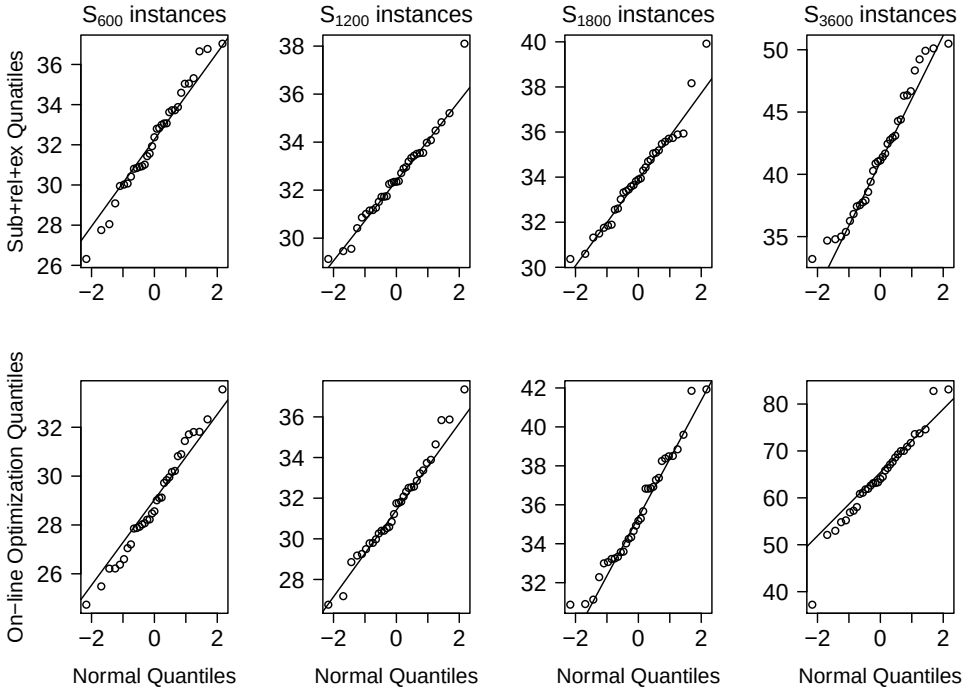


Figure B.2: Normal quantile plots of cost results on S_* scenarios.

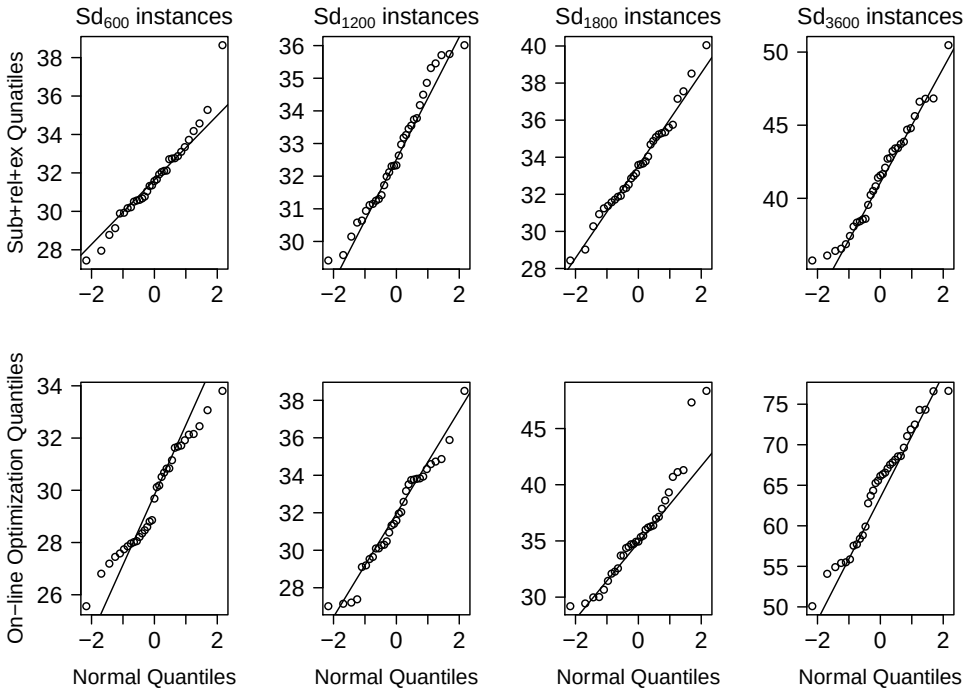


Figure B.3: Normal quantile plots of cost results on Sd_* scenarios.

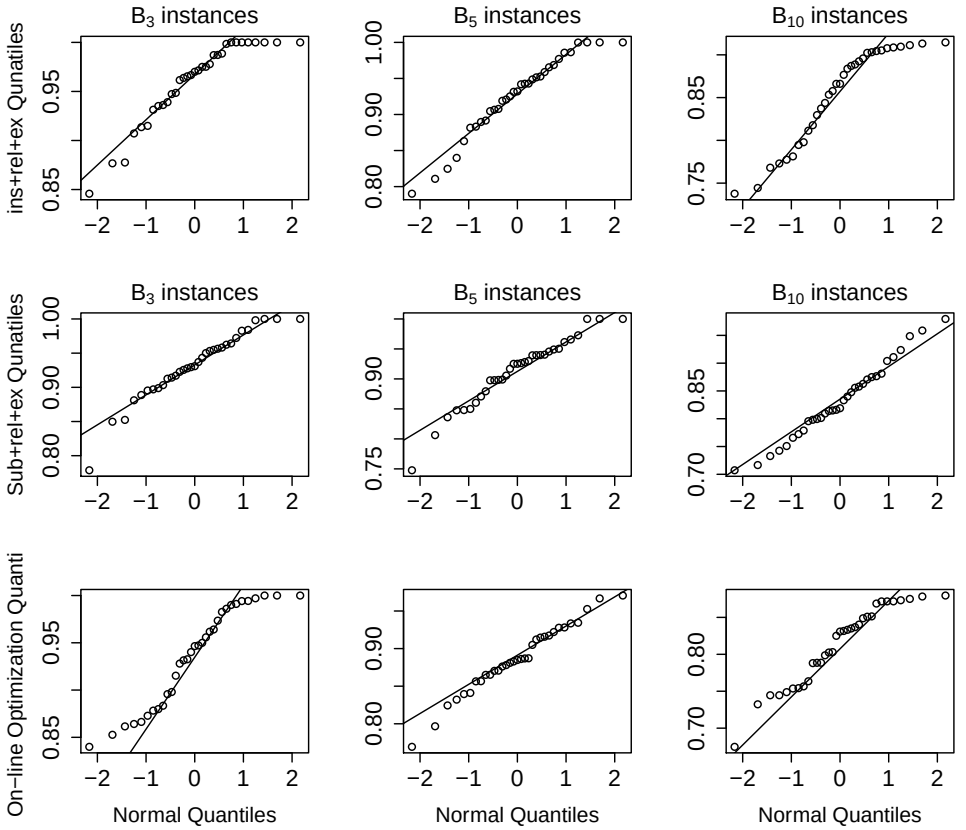


Figure B.4: Normal quantile plots of robustness results on B_* scenarios.

Afterword

In this thesis, we compare two different computational approaches to vehicle routing in an uncertain environment. While both approaches are based on standard *Operations Research* (OR) algorithms, one of them applies these algorithms in the usual centralized manner, while the other one applies them in a distributed way using a multi-agent model. Although we discuss various aspects of the two methods throughout the text, there is, however, an aspect that deserves further attention. Multi-agent modeling has been developed as a sub-field of *Artificial Intelligence* (AI), and as such, it aims to produce computer solutions that exhibit intelligent behavior. Developing an agent-based solution for vehicle routing suggests that this solution method is in some sense intelligent in routing vehicles in an uncertain world. But we have just claimed that the agent-based method of this book relied on the distributed application of OR algorithms. In what sense, then, is the distributed application of OR algorithms intelligent?

When we look around in our world searching for intelligence, we often find that intelligent behavior is the result of some network activity. The human brain, our primary source of intelligence, is a network of individual brain cells, each activating individually in response to stimuli on incoming connections. The braincells themselves can be described by simple functions, but the behavior emerging from the interactions of these simple functions is so complex that it is still beyond comprehension. Other examples are human or animal societies, one of the motivators behind multi-agent modeling. These societies consist of numerous individual specimen, whose actions are determined by individual decisions based on interactions with fellow society members. Such a society is often modeled as a network of agents, where the links in the network represent interactions between the agents. While an individual specimen might be rather simple, like an ant for example, the overall behavior of the society can still be quite complex, and indeed intelligent, like a foraging ant colony. In short, behind an observed intelligent behavior often lies a network of interacting rather simple entities. It seems like intelligence is not in the individuals, but it is in the network.

How does this relate to our original question of the intelligence in distributed vehicle routing solutions? Obviously, individual agents that model a small part of a vehicle routing problem, such as a truck or a parcel that needs to be transported, and that employ simple OR algorithms to achieve their goals are not intelligent themselves. A network of such simple entities, however, might behave intelligently in certain circumstances. While we are not making any claims in this respect here, and it is also not scientifically analysed in this thesis, we believe that making distributed models for any kind of problem is the first step towards developing an intelligent solution to that problem.

Acknowledgement

During my PhD and, in general, my years in the Netherlands, I was motivated by and received help from numerous colleagues and friends. I wish to thank them now.

To start at the very beginning, I have to thank Hans Abbink and Peet van Tooren to offer me a job at Almende. But not only that, because they not only made it possible for me to come to the Netherlands, they also spent numerous hours explaining me their unconventional ideas about agent systems, software engineering, artificial intelligence, the universe, and everything. Playing „go" with Hans overnight, or spending a whole day discussing rather philosophical topics with Peet, these are among the most valued experiences in my life.

And then there are the colleagues at Almende. Kees, first of all, then Jan Peter and Stefan, who were there from the beginning, and Adriaan, and Xiao Yu (my comrades in the hurdles of earning a PhD), Alfons, Anne, Andries, Duco, and of course Judith. They, and the others, made working at Almende an exciting experience and a great pleasure. I shall never forget the long Friday nights with Kees, drinking beers and playing pool. I thank them all for their hospitality and the buoyant atmosphere of the office. Although we were colleagues only for two years, Gerald de Jong is among my most influential acquaintances. He always stood two feet on the ground, even when all else seemed to go insane, and I always enjoyed discussing with him the hot topics of the day, professional or political. Gerald, I hope we can stay in touch, and continue to have fun together!

In my early years at Almende, working on an agent system for transportation, I had the incredible luck of cooperating with Mathijs de Weerdt. He was finishing his PhD at the time, and he spent a half year at Almende in Rotterdam before starting his academic career in Delft. Later on, when I joined him in Delft, he became my daily supervisor and in the end my co-promotor. I dare say that all I know about research, I have learnt from him. How to analyse problems, how to ask the right questions, how to search the literature, and how to write the articles; he helped me all along the way. I owe Mathijs a dept of gratitude for believing in me and supporting me all through these years.

From the Delft University of Technology, I want to thank, first of all, prof. Cees Witteveen, who provided guidance whenever I needed it. Most importantly he helped me in finding my topic in the beginning (and at half time again), and in writing a book about my findings, this thesis. Cees, I do not think I could have found a better professor for my PhD than you. I thank you for your rigorous scientific attitude, and your jokes! And the same goes to the other colleagues in the group: Adriaan, Chetan, Jonne, Pieter, Léon, Sicco, Yingqian, Marijn, Renze, Thomas, and Hans.

In most of the time I spent on my PhD, I was cooperating with colleagues from the Erasmus University in Rotterdam within the framework of the DEAL project. In the first few years, I had the joy of working with Hans Moonen and Elfriede Krauth. We spent countless meetings and train-rides together, not to mention the Riezlern ski-conferences for which I also thank Rene de Koster and Steef van der Velde. These were really a highlight in the normal routine, and they brought me closer to understand the field of transportation. I count myself extremely lucky for the chance of cooperating with Jordan F. Srour. It was always a pleasure working with her, and without her contribution, I could not have written this book. But it is not only our scientific work I set great store by. I also appreciate all the times we spent together during the summer school in Denmark, or skiing in Riezlern.

Finally, on a more personal note, I would like to thank Veronika László for becoming a friend of mine. We met by accident in a shop in 2003, and we quickly realized that we talk a common language. She and Károly Tóth helped me in the early days to settle down, and later on we became inseparable. Even with 1400 kilometers in between, it still feels like we are neighbors.

I am also grateful to another friend, Ábel Gyimesi, for moving to the Netherlands and marrying Mirjam van den Bos. It was on their wedding, when I met my wife, Krisztina Gábor. Krisztina stood by me almost from the beginning of my PhD, and she showed me a good example by finishing hers in time. She helped me by providing an outsider's point of view, and she proof-read parts of the text. I thank her for being with me, for giving purpose to my days, for giving me Menta, our daughter, and for expecting our second child!

Last, but nor least, I wish to thank my parents, my brother, and my grandmother for supporting me during my time in the Netherlands.

Tamás Máhr
Budapest, August 2011

Summary

Routing vehicles to pick up and deliver goods between locations is a frequently occurring problem in our globalized world. Researchers have studied several different variants of this problem and numerous different solution methods have been developed to solve such problems. Most of these methods produce a plan that, if followed by the vehicles, ensures that all orders are delivered in time.

Recently, problems in which not all information is known when planning has to start have become the focus of research. In these problems, the initially computed plan has to be updated during execution, when new information, e.g. a new order, is revealed. In this thesis, we call such problems vehicle-routing problems with uncertainty. Here, uncertainty stands for any unforeseen events that happen during the delivery of goods, and that makes an update of the plan necessary.

In this thesis, the main focus is on the study of a real-world transportation problem with uncertainties, and on the comparison of a centralized and a distributed solution approach in the context of this problem. We formalize the real-world problem, and provide a general framework to extend it with different sources of uncertainty. We apply computer simulations to compare the two solution approaches, and provide experimental results for three uncertainty types. In release-time uncertainty instances, new orders arrive at random times during the execution of previously planned orders. In service-time uncertainty cases, the time spent by the trucks at pick-up or delivery locations are defined according to a random distribution. In truck-breakdown uncertainty instances, some of the trucks break down at random times. Finally, we also examine cases with different combinations of these uncertainties.

When discussing our results, we seek answers for four research questions. The first one, Research Question 1 raises the issue of formalizing uncertainty in transportation problems.

> *How should uncertainties in transportation problems be formalized, and what kind of framework should be developed to analyze such problems?*

A formal description of uncertainties in transportation is important for a proper analysis of the problem; it provides a common understanding for the discussions, and it facilitates the development of planning methods that handle uncertainties better. To this end, this thesis provides an uncertainty framework as an extension to classical transportation-problem formulations. For example, a pick-up and delivery problem can be extended with different sources of uncertainty, such as truck breakdowns, or service-time variations. With the help of this uncertainty framework, any previously defined static problem can be turned into a dynamic transportation problem. Thanks to the separated definition of the static transportation problem and the sources of dynamism, the

model facilitates the reuse of transportation problems and incident definitions. Although other formalizations are also possible, our proposed model has the interesting property that it allows the definition of uncertainties that are arbitrary combinations of other uncertainties. In this way, complex real-world problems can be modeled by a composition of basic uncertainty types.

Related to this uncertainty framework, Research Question 1a asks for an appropriate measure of the effect of uncertainties on transportation planning methods.

How can we quantify the effects of uncertainties on planning methods?

There are multiple ways one can think about the effects of uncertainties in transportation problems. Our answer to this question is a new measure of robustness that quantifies the degradation of the (cost-based) performance of a transportation-planning method under uncertainty. Our robustness measure is a dimension-less, normalized measure that is based on the ratio of the method's average cost-based performance in the static vs. uncertain instances.

$$r(S) = \frac{1}{|S|} \sum_{i \in S} \min \left(1, \frac{c(\hat{i})}{c(i)} \right),$$

where S is a set of problem instances, $c(i)$ is the cost of transportation on instance i, and $\hat{\imath}$ is the static variant of i.

It is practically undoable to study and compare inherently different approaches to solving transportation problems in practice, therefore, we need some kind of environment to evaluate such approaches. Research Question 1b enquires about a proper simulation framework for such evaluation.

What are the requirements of a simulation test-bed that is used to evaluate planning methods in transportation problems with uncertainty?

During our investigations we concluded that a simulation framework of transportation problems with uncertainties should meet the following three requirements to provide a fair comparison of on-line planning approaches. It should be able to simulate unforeseen events; it should separate the planning algorithm and the simulation of the execution of the plans; and it should simulate the execution of the plans in real time. The first requirement is obvious, and the other two requirements aim to ensure that different kinds of planning methods can be compared, and that the computation time of these methods are considered by the framework.

A good simulation also requires realistic data on jobs, locations, etc. Since benchmark data for transportation problems is rare, we based our comparison study on instances generated from a real dataset. We were fortunate to have access to the operational data of a mid-sized Dutch transportation company. We generated several different problem instances that reflect the every-day uncertainties experienced by the planners of the company. The prepared instances contained release-time, two types of service-time, and truck-breakdown uncertainty cases, and combinations of these uncertainties. To help subsequent studies on dynamic transportation methods, we published the generated release-time uncertainty instances (Srour et al., 2010), and contributed them to MIPLIB, an electronically available library of both pure and mixed integer programs.

The above discussed three research questions investigate how transportation planning methods under uncertainty can be, and should be studied. Our main purpose of studying these questions is to facilitate the examination of the main topic of this thesis, the competition of centralized and distributed planning methods for transportation problems with uncertainty. This is outlined by Research Question 2 as follows.

> *How do uncertainties in transportation problems affect centralized and distributed solution methods of such problems?*

This question is studied in this thesis to the extent of the three different types of uncertainties mentioned above: release-time, service-time, and truck-breakdown uncertainties. We compared the performance of three different methods using simulation. An on-line optimization method and two agent-based heuristics (a classical, and our new relocation heuristic) were compared in static problem instances, instances with pure uncertainties, and instances with mixed uncertainties.

In the tests of pure-uncertainty scenarios, we found that above certain levels of uncertainty, in the cases of release-time and service-time uncertainties, the agent methods had a better cost performance than the optimization-based method. To be precise, when less than half of the orders arrived during the day, the optimization performed significantly better. When more than 75% of the orders arrived during the day, and therefore could not be planned for in advance, the agent methods performed significantly better. Between the two levels (50%-75%) the two approaches produced statistically indistinguishable ($p > 0.05$) results. When all orders were static, but the service times were varied by a maximum of 20 minutes, the on-line optimization produced better results, but when the variation was higher than that, the agent methods were more cost effective. Interestingly, in the truck-breakdown scenarios no such tipping point was found. The on-line optimization was consistently better than the agent-methods at any (tested) level of truck breakdowns.

To see how the methods perform when more than one source of uncertainty is present, we combined the high uncertainty cases from the three uncertainty types. Based on the conclusions of the pure-uncertainty scenarios, we combined two scenarios that favoured the agent methods (with the highest level of release-time, and service-time uncertainty), and one that favoured the optimization method (with the highest level of truck-breakdown uncertainty). In the combination of the release-time and service-time scenarios, the agent methods prevailed, but in the notable combination of release-time and truck-breakdown uncertainties the three methods performed equally. In the combination of the service-time and truck-breakdown uncertainties the agent methods were more cost effective. These results suggest that while the maximum level of release-time and truck-breakdown uncertainties counter-balanced the effects of each other, the effect of an approximately one-hour service-time variation was stronger than the effect of 10 truck breakdowns. Based on these results, it is not surprising that in the combination of all three uncertainty types, the agent methods were significantly better than the on-line optimization method.

In addition to the classical cost measure, we also evaluated the three methods using our robustness measure. The agent methods were found to be more robust than the on-line optimization in all (pure and combined) scenarios, except in the truck-breakdown uncertainty cases. In short, our simulation results show that lightweight distributed local heuristics are competitive with

centralized optimization methods, and in certain circumstances they can even outperform the centralized method both in terms of cost, and our robustness measure.

To summarize, this thesis contributed to the state of the art of transportation planning by providing i) an uncertainty framework for transportation problems with uncertainties, ii) a robustness measure to quantify the effect of uncertainties on planning methods, iii) a proper simulation framework for studying transportation problems with uncertainties, and iv) a comparison study of centralized and distributed planning methods using this simulation framework. This work is aimed towards providing more reliable, more robust planning methods, and it intends to encourage and facilitate follow-up research into this direction.

Samenvatting

Het routeren van voertuigen voor het bezorgen van goederen is een veel voorkomend probleem in onze geglobaliseerde wereld. Onderzoekers hebben verschillende varianten van dit probleem bestudeerd en verscheidene oplossingsmethodes zijn ontwikkeld om deze problemen op te lossen. Het merendeel van de methodes produceert een plan dat, indien gevolgd door alle voertuigen, ervoor zorgt dat alle opdrachten (*orders*) op tijd bezorgd worden.

Recentelijk zijn problemen waarin niet alle informatie bekend is bij aanvang van de planning onderwerp van onderzoek geworden. Bij deze problemen kunnen eerder berekende plannen worden aangepast tijdens de uitvoeringsfase, zodra nieuwe informatie, bijvoorbeeld in de vorm van een nieuwe opdracht, beschikbaar komt. In dit proefschrift noemen we dergelijke problemen routeplanningsproblemen met onzekerheid (*vehicle-routing problems with uncertainty*). Met onzekerheid bedoelen we hier willekeurige gebeurtenissen die op kunnen treden bij het bezorgen van goederen en die een herziening van het plan noodzakelijk maken.

De nadruk in dit proefschrift ligt op de studie van realistische transportproblemen met onzekerheid, en op de vergelijking tussen centrale en gedistribueerde oplossingsmethodes in de context van dit probleem. We presenteren een formeel model van een realistisch transportprobleem, en geven een algemeen raamwerk om dit model uit te breiden met verschillende bronnen van onzekerheid. We gebruiken computer simulaties om de twee benaderingen te vergelijken, en presenteren resultaten van experimenten met drie soorten onzekerheden. Bij opdrachtsonzekerheid (*release-time uncertainty*), komen op willekeurige tijdstippen tijdens de uitvoering van ingeplande orders nieuwe opdrachten binnen. In gevallen van uitvoeringsonzekerheid (*service-time uncertainty*) wordt de tijd die de vrachtwagen nodig heeft voor het inladen en uitladen bepaald door een kansverdeling. Bij voertuigonzekerheid (*truck-breakdown uncertainty*) kunnen vrachtwagens op willekeurige tijdstippen uitvallen. Tenslotte bekijken we ook gevallen waarbij verschillende combinaties van deze onzekerheden mogelijk zijn.

Bij de bespreking van onze resultaten probreren we antwoord te geven op onze vier onderzoeksvragen. De eerste, Onderzoeksvraag 1, betreft het formaliseren van onzekerheden in transportproblemen.

> *Hoe zouden onzekerheden in transportproblemen geformaliseerd moeten worden, en wat voor raamwerk moet ontwikkeld worden om dergelijke problemen te kunnen analyseren?*

Een formele beschrijving van onzekerheden in transport is belangrijk voor een gedegen analyse van het probleem; het verschaft een gemeenschappelijke basis voor de discussie van het probleem,

en het faciliteert de ontwikkeling van planningsmethodes die beter met onzekerheden om kunnen gaan. Daartoe wordt in dit proefschrift een raamwerk gepresenteerd waarmee onzekerheden kunnen worden gemodelleerd als uitbreidingen op klassieke formuleringen van transportproblemen. Een ophaal-en-bezorgprobleem (*pickup and delivery problem*), bijvoorbeeld, kan uitgebreid worden met verscheidene bronnen van onzekerheid, zoals uitvoeringsonzekerheid en voertuigonzekerheid. Met behulp van dit onzekerheidsraamwerk kan ieder eerder geformuleerd statisch transportprobleem in een dynamisch transportprobleem veranderd worden. Dankzij de scheiding tussen de beschrijving van het statische transportproblemen en de bronnen van dynamiek, werkt het model hergebruik van transportprobleem definities en incident beschrijvingen in de hand. Ook al zijn andere manieren van formaliseren mogelijk, onze aanpak heeft de interessante eigenschap dat het toestaat om onzekerheden te definiëren als willekeurige combinaties van andere onzekerheden. Op deze wijze kunnen complexe problemen uit de realiteit gemodelleerd worden door de compositie van basale vormen van onzekerheid.

Met betrekking tot dit raamwerk vraagt Onderzoeksvraag 1a wat een geschikte maat is voor de effecten die onzekerheden op de transportplanning hebben.

Hoe kunnen we de effecten van onzekerheden op planningsmethodes kwantificeren?

Er zijn verschillende manieren om de effecten van onzekerheden op transportplanning te beschouwen. Ons antwoord op deze vraag is een nieuwe maat van robuustheid die de verslechtering kwantificeert van de (op kosten gebaseerde) prestaties van een transportplanningsmethode bij onzekerheid. Onze robuustheidsmaat is een eenheidsvrije, genormaliseerde waarde die is gebaseerd op de verhouding tussen de gemiddelde prestaties in statische vs. dynamische instanties.

$$r(S) = \frac{1}{|S|} \sum_{i \in S} \min\left(1, \frac{c(\hat{\imath})}{c(i)}\right),$$

waarin S een verzameling probleeminstanties is, $c(i)$ de kosten van transport voor instantie i, en $\hat{\imath}$ is de statische variant van i.

Het is vrijwel ondoenlijk om wezenlijk andere aanpakken voor transportproblemen met elkaar te vergelijken in de praktijk, zodat we dergelijke aanpakken in een andere omgeving moeten evalueren. Onderzoeksvraag 1b betreft een geschikt simulatie-raamwerk voor zo'n evaluatie.

Wat zijn de vereisten van een simulatie-omgeving die gebruikt wordt om planningsmethodes voor transportproblemen met onzekerheid te evalueren?

Gedurende ons onderzoek hebben we geconcludeerd dat een raamwerk voor simulatie van transportproblemen met onzekerheid aan de volgende drie eisen moet voldoen om een eerlijke vergelijking van *online* planningsaanpakken mogelijk te maken. Het moet onvoorziene gebeurtenissen simuleren; het moet het planningsalgoritme scheiden van de simulatie van de uitvoer van de plannen; en het moet de uitvoering van plannen in werkelijke tijd (*real time*) kunnen simuleren. De eerste vereiste is vanzelfsprekend, en de andere twee eisen proberen te verzekeren dat verschillende soorten planningsmethodes vergeleken kunnen worden, en dat bij de vergelijking rekening gehouden kan worden met de verschillende rekentijden die de verschillende aanpakken nodig hebben.

Voor een goede simulatie zijn ook realistische gegevens nodig van opdrachten, locaties, enz. Aangezien benchmark gegevens voor transportproblemen schaars zijn, hebben we onze vergelijking gebaseerd op instanties die zijn gegenereerd uit een verzameling werkelijke gegevens. We mochten ons gelukkig prijzen met toegang tot operationele gegevens van een middengroot Nederlands transportbedrijf. We hebben een aantal verschillende soorten instanties gegenereerd die de onzekerheden weerspiegelen waar de planners van het bedrijf dagelijks mee te maken krijgen. De geprepareerde instanties bevatten opdrachtsonzekerheid, twee soorten uitvoeringsonzekerheid, voertuigonzekerheid, en combinaties van deze onzekerheden. Om vervolgstudies naar dynamische transportmethodes te helpen, hebben we gegenereerde instanties met opdrachtsonzekerheid (Srour et al., 2010) gepubliceerd en bijgedragen aan de MIPLIB, een elektronisch verkrijgbare bibliotheek van zowel pure en mixed integer programma's (*pure* en *mixed integer programs*).

De hierboven besproken drie onderzoeksvragen onderzoeken hoe planningsmethodes voor transportproblemen met onzekerheid kunnen, en zouden moeten worden bestudeerd. Het voornaamste doel van de bestudering van deze vragen is om de centrale onderzoeksvraag van dit proefschrift te kunnen bestuderen, betreffende de competitie tussen gecentraliseerde en gedistribueerde planningsmethodes voor transportproblemen met onzekerheid. Dit is als volgt verwoord in Onderzoeksvraag 2.

> *Hoe beïnvloeden onzekerheden in transportproblemen gecentraliseerde en gedistribueerde oplossingsmethodes voor zulke problemen?*

Deze vraag wordt in dit proefschrift bestudeerd in de context van de drie eerder genoemde vormen van onzekerheid: opdrachtsonzekerheid (*release-time uncertainty*), uitvoeringsonzekerheid (*service-time uncertainty*), en voertuigonzekerheid (*truck-breakdown uncertainty*). We hebben de prestaties van drie verschillende methodes vergeleken in simulatie. Een *online* optimaliseringsmethode, en twee agent-(*agent)*gebaseerde heuristieken (een klassieke heuristiek en onze nieuwe herplaatsingsheuristiek) zijn vergeleken in statische probleem instanties, in instanties met enkelvoudige onzekerheden, en in instanties met gecombineerde onzekerheden.

In de experimenten met één type onzekerheid hebben we, in geval van opdrachts- en uitvoeringsonzekerheid, kunnen vaststellen dat de agent methodes een betere prestatie leveren dan de op optimalisatie gebaseerde methode. Om precies te zijn, indien minder dan de helft van de opdrachten dynamisch arriveerde, was de optimalisatie methode significant beter. Als meer dan 75% van de opdrachten in de loop van de dag binnenkwam, en dus niet van tevoren ingepland kon worden, dan deden de agent methodes het significant beter. Tussen de twee niveaus (50%-75%) produceerden de twee typen aanpakken statistisch gezien ononderscheidbare resultaten ($p > 0.05$). Als alle opdrachten statisch waren, maar de uitvoertijden konden met maximaal 20 minuten variëren, dan gaf de online optimalisatie betere resultaten, maar als de variatie hoger was, dan waren de agent methodes effectiever. Interessant genoeg was er in het geval van voertuigonzekerheid niet zo'n keerpunt aan te wijzen. De online optimalisatie was altijd beter dan de agent methodes voor ieder niveau van vrachtwagenuitval.

Om te bekijken hoe de methodes presteren als er meer dan één bron van onzekerheid aanwezig is, hebben we de drie types onderzekerheid gecombineerd, met ieder type op een hoog niveau van onzekerheid. Op basis van de conclusies van de scenario's met een enkel

type onzekerheid hebben we twee scenario's gecombineerd die de agent methodes zouden moeten bevoordelen (met hoge opdrachts- en uitvoeringsonzekerheid), en één die in het voordeel was van de optimalisatie methode (met het hoogste niveau van vrachtwagenuitval). Voor de combinatie van opdrachts- en uitvoeringsonzekerheid deden de agent methodes het beter, maar in de combinatie van opdrachtsonzekerheid en vrachtwagenuitval presteerden de drie methodes gelijkwaardig. Voor de combinatie van uitvoeringsonzekerheid en vrachtwagenuitval produceerden de agent methodes lagere kosten. Deze resultaten suggereren dat de effecten (op de planningsmethodes) van opdrachtsonzekerheid en vrachtwagenuitval tegen elkaar opwegen, terwijl de effecten van een variatie in de uitvoeringstijd van ongeveer een uur sterker waren dan de effecten van 10 verwachte uitgevallen vrachtwagens. Op basis van deze resultaten is het niet verrassend dat de agent methodes significant beter waren dan de optimalisatie methode voor de combinatie van de drie soorten onzekerheden.

Naast de klassieke kostenmaat hebben we de drie methodes ook geëvalueerd op basis van onze robuustheidsmaat. De agent methodes bleken robuuster dan de online optimalisatie in alle (enkelvoudige en gecombineerde) scenario's, behalve voor de gevallen van vrachtwagenuitval. In het kort: onze simulaties hebben aangetoond dat lichtgewicht, gedistribueerde heuristieken kunnen concurreren met gecentraliseerde optimalisatie methodes, en ze in bepaalde gevallen zelfs beter kunnen presteren, zowel op het gebied van kosten, als van robuustheid.

Om samen te vatten heeft dit proefschrift bijgedragen aan de *state of the art* van transportplanning met i) een raamwerk voor onzekerheid in transportproblemen, ii) een maat van robuustheid die de effecten van onzekerheid op planningsmethodes kan kwantificeren, iii) een gedegen simulatie-raamwerk voor het bestuderen van transportproblemen met onzekerheid, en iv) een vergelijkend onderzoek tussen gecentraliseerde en gedistribueerde planningsmethodes gebruik makend van de simulatie-omgeving. Dit werk is gericht op het leveren van meer betrouwbare, robuustere planningsmethodes, en het hoopt vervolgonderzoek in deze richting aan te moedigen en te faciliteren.

Curriculum Vitae

Tamás Máhr was born in Budapest, Hungary, on 13th October 1977. He attended local primary and high schools, and finished his secondary education in the Óbudai Gimnázium in 1996. Then, Tamás enrolled in the Budapest University of Technology, and studied software engineering with a specialization in telecommunication systems. He received his diploma in 2001, and became a MSc. in Technical Informatics.

After finishing his MSc. studies, he stayed at the Department of Telecommunication and Media Informatics at the Budapest University of Technology and Economics, and started a PhD curriculum. He investigated bandwidth-brokering technologies in differentiated-services networks under the supervision of prof. Tamás Henk for two years. In 2003, he discontinued his PhD studies in Hungary.

In March 2003, Tamás moved to Rotterdam, the Netherlands, and started a software engineering job at Almende BV, a small research company led by Hans Abbink and Peet van Tooren. At Almende, Tamás investigated multi-agent systems, distributed machine-learning methods, and other related fields of distributed artificial intelligence. A year later, he received a guest position as a PhD student in the group of prof. Cees Witteveen at the Department of Software Technology of Delft University of Technology. His research on agent-based solution methods for transportation problems with uncertainties was supervised by prof. Cees Witteveen, and dr. Mathijs M. de Weerdt from the university, and dr. Alfons Salden from Almende BV.

In 2009, Tamás moved back to Budapest, Hungary, and started his current job as a project manager at AITIA International Inc., a company developing multi-agent simulation tools, and intelligent data-processing algorithms.

List of Publications

Máhr, T. and de Weerdt, M. M. (2004). Distributed agent platform for advanced logistics. In *Proceedings of the Belgium-Dutch Conference on Artificial Intelligence (BNAIC-04)*, pages 395–396. BNVKI

Máhr, T. and de Weerdt, M. (2005). Distributed agent platform for advanced logistics. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 155–156. ACM Press, New York, NY, USA. ISBN 1-59593-093-0. Accepted as poster

Máhr, T., Valk, J., and van Tooren, P. (2005). A logistical multi-agent system. In *Proceedings of International Conference on Self-Organization and Adaptation of Multi-agent and Grid Systems: Emerging Works on Autonomic Informatics*. SOAS'2005

Máhr, T. and de Weerdt, M. M. (2006). Multi-attribute vickrey auctions when utility functions are unknown. In Schobbens, P.-Y., Vanhoof, W., and Schwanen, G., editors, *Proceedings of the Belgium-Dutch Conference on Artificial Intelligence (BNAIC-06)*, pages 221–227. BNVKI. ISBN 1568-7805

Máhr, T., de Weerdt, M. M., Srour, J., and Zuidwijk, R. (2006). Incorporating expert knowledge in decision support for logistics: Balancing costs and customer relations. In *Proceedings of the 9th TRAIL congress: TRAIL in Motion*, pages 217–232. TRAIL Research School

Máhr, T. and de Weerdt, M. (2007). Auctions with arbitrary deals. In Mařík, V., Vyatkin, V., and Colombo, A., editors, *HoloMAS 2007*, volume 4659 of *LNAI*, pages 37–46. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-74478-8

Srour, J., Máhr, T., de Weerdt, M. M., Zuidwijk, R., and Moonen, H. (2008). *Smart Business Networks: A new business paradigm*, chapter Performance Evaluation within a Networked Enterprise: Balancing Local Objectives and Network Relations, pages 279–304. ERIM Electronic series. SBNi. ISBN 978-90-5892-159-8. See `http://www.erim.eur.nl/ERIM/Research/Centres/SBNi/SBNi_Updates/Newsetail?p_item_id=5120098&p_pg_id=93`

Máhr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2008). Agent performance in vehicle routing when the only thing certain is uncertainty. In *Proceedings of the workshop on Agents in Traffic and Transportation (ATT)*

Srour, F. J., Máhr, T., de Weerdt, M. M., and Zuidwijk, R. A. (2010). MIPLIB truckload PDPTW instances derived from a real-world drayage case. In *ERIM report series research in management Erasmus Research Institute of Management*. Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam (ERIM is the joint researchinstitute of the Rotterdam School of Management, Erasmus University and the Erasmus School of Economics (ESE) at Erasmus

University Rotterdam)
Máhr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2010). Can agents measure up? a comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research Part C: Emerging Technologies*, 18(1):99 – 119. ISSN 0968-090X. Information/Communication Technologies and Travel Behaviour; Agents in Traffic and Transportation
Máhr, T., Srour, J., and de Weerdt, M. M. (2011). Using simulation to evaluate how multi-agent transportation planners cope with truck breakdowns. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control*. IEEE

TRAIL Thesis Series

The following list contains the most recent dissertations in the TRAIL Thesis Series. For a complete overview of more than 100 titles see the TRAIL website: www.rsTRAIL.nl.

The TRAIL Thesis Series is a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics.

Máhr, T., *Vehicle Routing under Uncertainty*, T2011/11, September 2011, TRAIL Thesis Series, the Netherlands

Pel, A.J., *Transportation Modelling for Large-scale Evacuations*, T2011/10, July 2011, TRAIL Thesis Series, the Netherlands

Zheng, F., *Modelling Urban Travel Times*, T2011/9, July 2011, TRAIL Thesis Series, the Netherlands

Vlassenroot, S.H.M., *The Acceptability of In-vehicle Intelligent Speed Assistance (ISA) Systems: from trial support to public support*, T2011/8, June 2011, TRAIL Thesis Series, the Netherlands

Kroesen, M., *Human Response to Aircraft Noise*, T2011/7, June 2011, TRAIL Thesis Series, the Netherlands

Nielsen, L.K., *Rolling Stock Rescheduling in Passenger Railways: applications in short-term planning and in disruption management*, T2011/6, May 2011, TRAIL Thesis Series, the Netherlands

Qing, O., *New Approach to Fusion of Heterogeneous Traffic Data*, T2011/5, May 2011, TRAIL Thesis Series, the Netherlands

Walta, L., *Getting ADAS on the Road: actors' interactions in Advanced Driver Assistance Systems deployment*, T2011/4, April 2011, TRAIL Thesis Series, the Netherlands

Lin, S., *Efficient Model Predictive Control for Large-Scale Urban Traffic Networks*, T2011/3, April 2011, TRAIL Thesis Series, the Netherlands

Oort, N. van, *Service Reliability and Urban Public Transport Design*, T2011/2, April 2011, TRAIL Thesis Series, the Netherlands

Mahmod, M.K.M., *Using Co-Operative Vehicle-Infrastructure Systems to Reduce Traffic Emissions and Improve Air Quality at Signalized Urban Intersections*, T2011/1, March 2011, TRAIL Thesis Series, the Netherlands

Corman, F., *Real-Time Railway Traffic Management: dispatching in complex, large and busy*

railway networks, T2010/14, December 2010, TRAIL Thesis Series, the Netherlands

Kwakkel, J., *The Treatment of Uncertainty in Airport Strategic Planning*, T2010/13, December 2010, TRAIL Thesis Series, the Netherlands

Pang, Y., *Intelligent Belt Conveyor Monitoring and Control*, T2010/12, December 2010, TRAIL Thesis Series, the Netherlands

Kim, N.S., *Intermodal Freight Transport on the Right Track? Environmental and economic performances and their trade-off*, T2010/11, December 2010, TRAIL Thesis Series, the Netherlands

Snelder, M., *Designing Robust Road Networks: a general design method applied to the Netherlands*, T2010/10, December 2010, TRAIL Thesis Series, the Netherlands

Hinsbergen, C.P.IJ. van, *Bayesian Data Assimilation for Improved Modeling of Road Traffic*, T2010/9, November 2010, TRAIL Thesis Series, the Netherlands

Zuurbier, F.S., *Intelligent Route Guidance*, T2010/8, November 2010, TRAIL Thesis Series, the Netherlands

Larco Martinelli, J.A., *Incorporating Worker-Specific Factors in Operations Management Models*, T2010/7, November 2010, TRAIL Thesis Series, the Netherlands

Ham, J.C. van, *Zeehavenontwikkeling in Nederland: naar een beter beleidsvormingsproces*, T2010/6, August 2010, TRAIL Thesis Series, the Netherlands

Boer, E. de, *School Concentration and School Travel*, T2010/5, June 2010, TRAIL Thesis Series, the Netherlands

Berg, M. van den, *Integrated Control of Mixed Traffic Networks using Model Predictive Control*, T2010/4, April 2010, TRAIL Thesis Series, the Netherlands

Top, J. van den, *Modelling Risk Control Measures in Railways*, T2010/3, April 2010, TRAIL Thesis Series, the Netherlands

Craen, S. de, *The X-factor: a longitudinal study of calibration in young novice drivers*, T2010/2, March 2010, TRAIL Thesis Series, the Netherlands

Tarau, A.N., *Model-based Control for Postal Automation and Baggage Handling*, T2010/1, January 2010, TRAIL Thesis Series, the Netherlands

Knoop, V.L., *Road Incidents and Network Dynamics: effects on driving behaviour and traffic congestion*, T2009/13, December 2009, TRAIL Thesis Series, the Netherlands

Baskar, L.D., *Traffic Control and Management with Intelligent Vehicle Highway Systems*, T2009/12, November 2009, TRAIL Thesis Series, the Netherlands

Konings, J.W., *Intermodal Barge Transport: network design, nodes and competitiveness*, T2009/11, November 2009, TRAIL Thesis Series, the Netherlands

Kusumaningtyas, I., *Mind Your Step: exploring aspects in the application of long accelerating moving walkways*, T2009/10, October 2009, TRAIL Thesis Series, the Netherlands

Gong, Y., *Stochastic Modelling and Analysis of Warehouse Operations*, T2009/9, September 2009, TRAIL Thesis Series, the Netherlands

Notes