MASTER

Solving the stochastic vehicle routing problem with service requirements in the DAIPEX project using adaptive large neighborhood search

Creemers, D.J.A.

Award date:
2016

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Solving the Stochastic Vehicle Routing Problem with Service Requirements in the DAIPEX project using Adaptive Large Neighborhood Search

Daan Creemers

supervised by

Prof.dr.ir. Wim Nuijten

19-12-2016

Public version

QUINTIQ

TU/e Technische Universiteit Eindhoven University of Technology

Contents

Abstract

Transportation companies face the challenge that their day-to-day transportation execution does not conform to the transportation plan that they made in advance. To a large extent this is caused by the fact that the software that aids in the creation of transportation plans, does not sufficiently take the real-world complexity the transportation company is faced with into account. As a consequence the transportation plans that are generated can lead to violated time windows, unnecessary delays, and underutilized transportation capacity. The real-world complexity of transportation planning is caused by the high level of detail that is required to get executable plans, the size of the instances as found in reality, and the large volumes of data that must be collected and processed to gather the information required to create the planning. A particular source of detail is stochasticity and time dependency of travel times and service times, which is the area where this thesis focuses on.

The work reported on in this thesis is done as part of the DAIPEX project. This project is funded by DINALOG and its goal is to develop algorithms and software that can handle time-dependent, stochastic, planning problems, based on high-volume information. In this thesis we concentrate on developing algorithms that generate transportation plans that include time-dependent and stochastic travel times and service times. We use information coming from data aggregation algorithms as developed by the IE&IS group of Eindhoven University of Technology (TU/e), where we also developed functionality that supplements that information there were it is missing and use it in a way that better represents reality.

We solve the DAIPEX Transportation Planning Problem (DTPP) where customers have service level requirements. A service level requirement of 95% expresses that the customer should be serviced on time with 95% certainty. To solve the DTPP we propose three techniques which we use in an Adaptive Large Neighborhood Search approach. The first technique uses time window slack where we reserve time at the end of a time window. The second technique overestimates travel times in different ways. We show that using time window slack or travel time slack gives better transportation plans compared to plans in which the expected travel time is used. We also show that determining the right level of overestimating is hard. The third technique explicitly considers stochasticity in the algorithms which gives further improvements. A computational study shows what improvements can be obtained under which scenarios when considering stochastic travel times. Furthermore, we developed a way to calculate at which time a driver should depart from the depot to best meet the customers service requirements.

"The art of programming is the art of organizing complexity, of mastering multitude and avoiding its bastard chaos as effectively as possible."

— Edsger W. Dijkstra

Acknowledgments

This master thesis is the last part of my study Computer Science and Engineering at Eindhoven University of Technology (TU/e). The project is part of the Data and Algorithms for Integrated Transportation Planning and Execution (DAIPEX) project, created by DINALOG, which is a collaboration project between Quintiq, TU/e, TomTom and four transportation companies. While writing this thesis I received a lot of support from several people. I would like to thank all of them, in particular the following.

First of all, I would like to thank my graduation supervisor Wim Nuijten for his support, enthusiasm and motivation. His thoughtful guidance and critical comments helped me during my research. Next I would like to thank Dan Roozemond and Mark Lekkerkerker for our insightful discussions.

I would also like to thank Ludwig van den Ouweland, Lieke Schreurs and Edwin de Jong for their support throughout the project. I would also like to thank my colleagues at Quintiq Products for welcoming me.

Last but not the least, I would like to thank my parents, family and friends for their continuous support and encouragement throughout the period of my study and while writing this thesis.

Daan Creemers
Eindhoven, November 2016

1

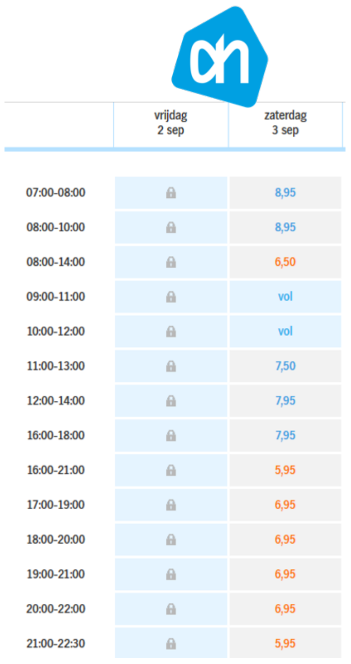
Introduction

1.1 Motivation

Uncertainty is a growing concern in the field of Supply Chain Planning and Optimization. It may arise from several sources including travel times in vehicle routing, processing times in manufacturing and changing demands in energy supply planning. While calculated plans look good if no uncertainty is taken into account, during execution of a plan companies are faced with disturbances making the plan suboptimal or even infeasible. An important issue there is that some decisions have a big impact if they cannot be executed as planned, while for other decisions the inability to execute them has limited to no impact. In vehicle routing we see that planning to deliver an order just before the end of the time window can be optimal in terms of costs, but due to uncertainty in travel and service times delivery of this order can suddenly be too late. If the time window is hard, there can be a high penalty on delivering late and if there is a lot of uncertainty on the arrival time you might be setting yourself up for failure.

When executing this project we also discovered the issue of driving time limits. Using a transportation plan with fixed travel times, we can guarantee that the driving time for every driver meets the daily driving limit set by the European Union. However, during execution of the transportation plan disturbances can lead to longer travel times than expected and a violated driving limit. Transportation companies are being fined when a driver that drove too long gets caught, and therefore, we want to take the driving limit into account when considering uncertainty of travel times.

A large part of the delivery business is concerned with time-boxed deliveries. Without agreeing on a delivery time window with a customer, chances are that the customer is not at home at the moment of delivery. We see a trend of services offered by companies where customers can choose their time window or where orders are delivered at the same day. Scheduled delivery, which is already provided by the supermarket chain Albert Heijn in The Netherlands as can be seen in Figure 1.1, is not only an advantage for the customer: transport companies can use this to prevent



	vrijdag 2 sep	zaterdag 3 sep
07:00-08:00	🔒	8,95
08:00-10:00	🔒	8,95
08:00-14:00	🔒	6,50
09:00-11:00	🔒	vol
10:00-12:00	🔒	vol
11:00-13:00	🔒	7,50
12:00-14:00	🔒	7,95
16:00-18:00	🔒	7,95
16:00-21:00	🔒	5,95
17:00-19:00	🔒	6,95
18:00-20:00	🔒	6,95
19:00-21:00	🔒	6,95
20:00-22:00	🔒	6,95
21:00-22:30	🔒	5,95

Figure 1.1: Scheduled delivery available with different costs at the Albert Heijn in The Netherlands.

repeated visits to a customer which are needed because of customer unavailability. Taking into account uncertainty becomes even more relevant with the increase of customers demanding smaller time windows. Ignoring uncertainty in this case can lead to dissatisfied customers because of late deliveries or penalty costs because of not meeting a pickup or delivery requirement of a company.

We also see the movement of time-boxed deliveries in the distribution and retail area. Stores demand that the inbound logistics from warehouses is delivered into small time-windows. For example because of limited storage space, or because the docking place can only be reached due to limited opening hours of city centers.

A simple solution to increase service reliability is increasing the number of routes used to serve all orders: less orders on a route there is less uncertainty which can cause service requirements violations. However, there are numerous disadvantages to this approach. First of all, the number of vehicles is limited in most transportation companies. Secondly, using an extra vehicle incurs extra costs because the driver and the vehicle has to be paid. Moreover, using more vehicles for the same number of orders causes the routes to be shorter. In many transportation companies drivers are contracted which means they are being paid for a full day. Planning routes for them for only half a day is thus expensive.

There are different methods to cope with stochastic travel times. We use the terms stochastic travel times and uncertain travel times interchangeably in this thesis. Planning algorithms which explicitly considering uncertainty by taking probability distributions into account should be able to generate transportation plans with less late deliveries to customers and less driving limit violations. If there is no information on the probability distribution of the travel times, we can still generate transportation plans which have value to a transportation company by using travel time slack and time window slack. When using travel time slack the planning algorithm assumes the travel on every arc takes longer than the expected travel time. In this way a driver will more likely arrive on time at a customer, with the disadvantage of a more expensive transportation plan because of waiting times or using more vehicles. When using time window slack the planning algorithm assumes the time window of every customer closes earlier than the actual time window. Using a transportation plan with time window slack, the driver may arrive at a customer before the time window opens meaning that the driver has to wait. The advantage is that even if the driver is delayed because of a traffic jam, he can still be on time at the customer. We will show that travel time slack and time window slack work under certain circumstances, while using the probability distributions of the travel times works better in general.

1.2 Quintiq and DAIPEX

This research has been conducted at Quintiq, a Supply Chain Planning and Optimization company with headquarters in Den Bosch. Quintiq sells software to solve scheduling and planning problems. Using optimization technology customers can lower their costs, produce more output, increase their profit or generate plans in less time. The research is done on behalf of Quintiq Products, a horizontal layer within Quintiq which builds industry-focused products like Logistics Planner, Company Planner, Demand Planner, Fleet & Crew Planner, Macro Planner, Scheduler, Service Planner and Workforce Planner. They continuously improve their planning and scheduling software, which is made available to the various business units and partners. One of the currently active fields of research is stochasticity and time-dependency of travel times which is the focus of this thesis.

Data and Algorithms for Integrated Transportation Planning and Execution, better known under the name DAIPEX, is a project of DINALOG, the Dutch Institute for Advanced Logistics. It is a collaboration project between Quintiq, Eindhoven University of Technology, TomTom and the four transportation companies H. Veldhuizen Transport, Jan de Rijk Logistics, Ewals Cargo Care, and Ernst Opus V. The goal is to develop generic, configurable planning software that takes into account stochasticity and time dependency of travel times based on high-volume information. The project considers integrating these stochastic dependencies in planning problems that arise in Cross Chain Control Centers (4C).

1.3 Research assignment

The research assignment of the project is formulated as follows:

- Do a relevant literature study in the area of vehicle routing under uncertainty with the focus on literature on Large Neighborhood Search (LNS) and on stochastic travel times.

- Provide a formal description of the DAIPEX Transportation Planning Problem (DTPP).
- Collect and construct real-life instances of the DTPP.
- Implement algorithms that find great solutions to the DTPP, where the optimizer can find solutions that under various circumstances provide the right trade-off between logistics costs and robustness of transportation plans.
- Investigate whether adaptive parameters (like adaptive penalties and adaptive destruct size) as used by Quintiq for deterministic vehicle routing problems also improves performance for the DTPP.
- Do a computational study which shows which algorithm works best under which circumstances.
- Do a computational study which shows the influence of components and parameter settings of the algorithms on performance.

1.4 Hypotheses

We started the project with a number of hypotheses. Besides the hypotheses listed below, we have defined more hypotheses while conducting research. These will be explained in Chapter 7.

Hypothesis 1 (Expected travel times)
Using the expected travel times when calculating a transportation plan gives bad plans in terms of service reliability.

Hypothesis 2 (Time window slack)
Using travel time slack by overestimating travel times works better than time window slack.

Hypothesis 3 (Travel time slack based on fixed percentage)
Building in slack by adding a certain percentage to the travel times will result in either many unmet service reliability requirements or an inefficient use of resources.

Hypothesis 4 (Travel time slack based on Standard Deviation)
Building in slack by overestimating the travel times based on the standard deviation of travel times works better than adding a fixed percentage to the travel time.

Hypothesis 5 (Travel time slack based on MAD)
Building in slack by overestimating the travel times based on the Mean Absolute Deviation from the Mean (MAD) works better than adding a fixed percentage to the travel time.

Hypothesis 6 (Sample Average Approximation)
Doing Sample Average Approximation works best in terms of quality, but takes more time and requires much more data of much higher quality.

1.5 Contributions

We have formally defined the DAIPEX Transportation Planning Problem (DTPP). Based on data provided by a transportation company we have constructed real-life instances of the DTPP with different circumstances. Moreover, we have defined the communication format used in the DAIPEX project to communicate travel time distributions.

We have used this communication format to obtain real-life travel time distributions coming from TomTom. We implemented a data completion algorithm capable to calculate realistic travel time distribution which are originally missing. Furthermore, we implemented a sampling mechanism allowing to get realistic simulations taking geographical dependencies into account.

We are capable to solve real-life transportation planning problems having time-dependent, stochastic travel and service times, where our optimizers can find solutions that under various circumstances provide the right trade-off between logistics costs and robustness of transportation plans. This is supported by our computational study where we show which solution approaches work best under which circumstances.

1.6 Structure

The remainder of this thesis is structured in the following way. Chapter 2 discusses the related work on vehicle routing and different solution approaches. The DAIPEX Transportation Planning Problem is formally introduced in Chapter 3. In Chapter 4 we describe the stochastic travel times and how we obtained these. Optimization algorithms to solve the DTPP are described in Chapter 5. In Chapter 6 we describe how to find the best start time of a route in order to satisfy all demands from customers but minimizing the costs. Chapter 7 presents a computational study showing the different strength and weaknesses of the various algorithms. We finish with the conclusion and future work in Chapter 8.

2

Related work

The DAIPEX Transportation Planning Problem (DTPP) reported on in this thesis is a generalization of the Vehicle Routing Problem (VRP). Much research has been conducted in this field, of which we give an overview in Section 2.1. The classical problem of vehicle routing assumes deterministic travel times between customers, but this can be extended by including stochastic input or having input which changes over time. In Section 2.2 we categorize optimization problems based on the quality and evolution of information. Section 2.3 discusses how optimization under uncertainty is dealt with in two other research fields to show that optimization under uncertainty is not only relevant for vehicle routing. Finally, in Section 2.4 we show different solution approaches to deal with optimization under uncertainty.

2.1 Vehicle Routing

The Vehicle Routing Problem (VRP) was defined by Dantzig and Ramser (1959) as a generalization of the Traveling Salesman Problem. The goal of the problem is to design an optimal set of routes for a fleet of identical capacitated vehicles all starting in a central depot such that each customer is visited exactly once while minimizing the total routing costs. Dantzig and Ramser propose a linear programming based heuristic which gives a near optimal solution for this NP-hard problem. Ever since their paper this problem has been extensively studied and various variants are introduced. Clarke and Wright (1964) present an algorithm which gives a big improvement over the algorithm proposed by Dantzig and Ramser. They start with a route for each customer and in each iteration the two routes which give the largest saving are merged. Although this algorithm can be performed by hand computation it is overtaken by many other algorithms.

Shaw (1998) introduced Large Neighborhood Search (LNS) for vehicle routing as an iterative technique 'which is compatible with state of the art Operations Research meta-heuristic methods, while being significantly simpler'. In each iteration of LNS a set of customers is removed from the solution and this set is inserted at other positions, therefore its acronyms 'Destroy and Repair' and 'Ruin and Recreate'.

The destroy and repair methods implicitly define the neighborhood of a solution. Shaw (1998) uses a branch and bound technique with constraint propagation and heuristics for variable and value selection to find the best position to insert the customers. They included Limited Discrepancy Search in the constraint-based tree search to speed up the time to solve problems. LNS is preferred over Local Search, because the latter has difficulties if the search space is pitted with local minima or is disconnected. This problem is partially alleviated by LNS, because a move by LNS is more far-reaching which allows to step over local minima or small disconnected regions. However, some regions in the search space can only be reached by moving through a large infeasible region of the search space. Increasing the neighborhood mitigates this problem, but can cause the search to be less inefficient as a result of jumping through the search space.

Adaptive Large Neighborhood Search (ALNS), introduced by Ropke and Pisinger (2006), extends the LNS heuristic by allowing multiple destroy and repair methods to be used within the same search. By assigning a weight to each method, a weighted selection is done to choose which method to use in the next LNS iteration. The weights are adjusted dynamically depending on the success in the iteration such that the heuristic adapts itself to the current instance and to the search phase. Besides including different destroy and repair methods, one can include the ordinary local search method to explore the close neighborhood around a solution from time to time.

Conijn (2013) solves the Vehicle Routing Problem with Stochastic Travel Times, but they assume time windows are hard. The DTPP assumes soft time windows, because we see in real-life that planners can decide to not satisfy a time window if this outweighs the saved costs for the reduced distance or for using a vehicle less.

2.2 Categorization of vehicle routing problems

In the previous section we have seen there are many variants of the Vehicle Routing Problem (VRP). In the classical definition by Dantzig and Ramser (1959) all travel and service times are deterministic and the demand request from customers is known before starting the optimization. These are strong assumptions which often do not apply in real-world applications. There are applications where new customer requests arrive on the go or the demand amount of a customer is only known when arriving at a customer.

Pillac, Gendreau, Guéret, and Medaglia (2013) distinguish two dimensions in real-world applications: *evolution* and *quality* of information. Evolution of information relates to information which changes over time, for instance a problem where new customer requests arrive while executing the plan. Quality of information relates to uncertain input because of uncertain environments or technical issues. Examples include only knowing a range for the demand amount or travel times being stochastic as a result of congestion. Note that stochastic travel times can be categorized in either of the two dimensions. Looking from one perspective we generate a plan with the stochastic travel times which we do not change during execution. Looking from the other perspective we generate a plan, but we keep in mind that during execution more information becomes available which allows to improve the plan.

The combination of information evolution and information quality gives four categories of optimization problems, presented in Table 2.1 by Pillac et al. (2013). When having *static and deterministic* information all information is known beforehand and this does not change during the execution of the generated solution. The classical VRP as discussed before is an example in this category. Other examples in the field of vehicle routing include, but are not limited to, Vehicle Routing with LIFO, where the order of loading and unloading goods is restricted to last-in-first-out (Carrabs, Cordeau, and Laporte (2007)), and Vehicle Routing with Soft Time Windows, where violating a time window is allowed taking into account that extra costs have to be paid (Taş, Dellaert, Van Woensel, and De Kok (2013)).

In *static and stochastic* problems the information does not change during execution of a plan, but there is uncertainty in the input data. The uncertain input is modeled by random variables and their realizations are revealed during the execution of the solution. A solution should be constructed based on the information that is available prior to the execution of the solution, for example based on the probability distribution of the random variables and on the input data which is fixed. The research reported on in this thesis belongs in this category: we assume we know the probability distributions of the travel times and we assume we know in advance which customers should be served and which demand they have. Two other categories of static and stochastic problems are stochastic demands and stochastic customers. In problems with stochastic demands the exact amount of the demand is only revealed when arriving at a customer (Lei, Laporte, and Guo (2011)). While generating a solution we should consider the stochastic demands such that a realization of the demands does not overload a vehicle. When having stochastic customers a customer has to be served with a certain probability (Bent and Van Hentenryck (2004)). Only after creating the solution it becomes known which customers can be omitted, while we still want efficient routes. In general the goal is to construct a solution which does not require replanning of actions during the execution of the solution. The solution should be valid during execution, because the problem is static and stochastic: the information does not change during execution of a solution. If the information changes over time, then real-time replanning can improve the solution. However, real-time replanning is more common in a dynamic setting as we will discuss next.

Both *dynamic and deterministic* problems and *dynamic and stochastic* problems are characterized by input not fully known a priori, but where the information is revealed during execution of the solution. The difference between the two problems is the information that is available a priori. While in dynamic and deterministic problems there is no information on the dynamically revealed information, in dynamic and stochastic problems there is stochastic information available in advance which can be exploited for the generation of a solution. For these types of problems technological support is required: positioning systems like GPS and real-time com-

		Information quality	
		Deterministic input	Stochastic input
Information evolution	Input known beforehand	Static and deterministic	STATIC AND STOCHASTIC
	Input changes over time	Dynamic and deterministic	Dynamic and stochastic

Table 2.1: Categorization of optimization problems by Pillac et al. (2013).

munication devices like mobile phones or centralized navigation products to allow the decision maker to communicate the improved solution to the vehicles. In the problem with uncertain travel times, real-time replanning can be beneficial when one vehicle ends up in traffic congestion while another vehicle is serving his customers expeditiously. We could then decide to pickup an order which was originally planned by another vehicle. This can even be more beneficial when it is not required to serve all customers, but when we can decide during execution if serving another customer is useful. These problems are also known as online planning problems, while static planning problem are known as offline planning problems. In this research we show that profit can be made when optimizing offline planning with uncertainty. We will show that some methods give faster good results, such that they can be used as online planning problems.

2.3 Optimization under Uncertainty

Besides vehicle routing there are other domains which are subject to uncertainty. We will shortly discuss two domains to see how uncertainty is handled in those problems.

In the field of manufacturing there are various types of uncertainty. Mula, Poler, Garcia-Sabater, and Lario (2006) distinguishes environmental uncertainty and system uncertainty. Environmental uncertainty includes uncertainties which are beyond the production process like demand uncertainty and supply uncertainty. System uncertainty on the other hand is related to the production process like production lead time uncertainty, failures of production systems and quality uncertainty. Mula et al. (2006) give a vast overview of manufacturing under uncertainty where the most frequently encountered modeling approach was stochastic programming, which we will discuss in Section 2.4. For example, Gupta and Maranas (2003) use a stochastic programming based approach to solve the problem of tactical planning of supply chains under demand uncertainty.

Another field which is subject to uncertainty is energy supply planning. Sources of uncertainty include markets, politics and technology. Lee (2014) propose a twostage stochastic program with recourse to solve the problem of coupling energy supply planning and energy supply chain design. Long-term decisions, like constructing new energy plants, are made based on stochastic input. After realization of the uncertain parameters the long-term decision can be countered by making short-term decisions, like the procurement of biomass and deciding on which material to process. Sharifzadeh, Garcia, and Shah (2015) show that seasonal and geographical uncertainties in biomass resources should be considered and different decisions should be made on the energy supply chain design. They develop a mixed integer (piece-wise) linear program to determine the optimal supply chain design under uncertainty. Considering uncertainty in this field is of importance because the design of the energy supply chain can have significant influence on the viability of the biofuel technology.

2.4 Solution approaches

When looking for solution approaches that can help solve the DTPP, the area of solving the Traveling Salesmen Problem (TSP) draws attention, as great progress has been made recently. While early algorithms could only solve small instances, current algorithms can handle millions of customers. Rego, Gamboa, Glover, and Osterman (2011) provides TSP heuristics to find solutions which are with a high probability only 1 to 3% away from the optimal solution in reasonable time. Although this provides hope to also get good results for the Vehicle Routing Problem with Time Windows (VRPTW), it turns out the VRPTW is more difficult. While in the TSP the stops on a route 'only' need to be sequenced, in the VRPTW there is a second planning decision of which stops to include on which route. As a result, problems with 100 customers are already difficult to solve. The DAIPEX Transportation Planning Problem is a very broad extension of the VRPTW amongst others by including driving regulations and stochastic time-dependent travel times. The combination of the VRPTW and the additional practical constraints has as result that we unfortunately cannot use the TSP heuristics.

One prominent practical constraint is about driving regulations. Driving regulations are in fact not often considered when solving the vehicle routing problem. Recently, Goel and Kok (2012) considered the truck driver scheduling problem in which a sequence of locations must be visited by a driver within given time windows under the US driving regulations. Their method can solve the truck driver scheduling problem in quadratic time, but only for US driving regulations. Vidal, Crainic, Gendreau, and Prins (2014) state that no polynomial algorithm is known for the truck driver scheduling problem under the more restrictive European (EU) driving regulations. Prescott-Gagnon, Desaulniers, Drexl, and Rousseau (2010) propose a Large Neighborhood Search algorithm using a column generation heuristic for the VRP with EU driving regulations. Columns are generated by tabu search that checks the feasibility with relation to driving regulations using a labeling algorithm.

Kok, Hans, Schutten, and Zijm (2010) also consider the EU driving regulations in their VRP, but extend this with traffic congestion by using time-dependent travel times. They use a dynamic programming heuristic by adding several dimensions, like remaining capacity, current time and the remaining travel time until a break must be scheduled, to each subproblem. Due to increased computation times they only expand a limited number of subproblems which have the lowest costs. We have not considered this approach, because adding stochastic travel times increases the complexity even more.

One component of the DTPP is the presence of uncertainty. Two major approaches to deal with optimization under uncertainty are Stochastic Programming and Robust Optimization, as discussed in Gorissen, Yanıkoğlu, and den Hertog (2015) and Evers (2013). They both focus on constructing solutions for which we expect they are good under many realizations of the data. This means that the goal is not to find the best solution possible by one realization of the uncertain data, but to find a good solution which is valid for a myriad of realizations of the uncertain data.

The main difference between the two approaches is the assumption that is made on the uncertain data. In Stochastic Programming the assumption is made that the true probability distribution of the uncertain data is known or can be estimated. In practice, probability distributions can be unknown or hard to obtain. On the other hand, Robust Optimization does not assume the probability distributions are known. Instead it assumes that uncertain parameters belong to a deterministic uncertainty set, that is, the uncertain parameters are unknown but bounded. Robust Optimization uses a min-max approach using the uncertainty set by optimizing the worst-case value of the optimization objective. The solution will satisfy all instances within the uncertainty set, and given all instances the solution will be optimal.

Early robust optimization algorithms provide conservative solutions, since the resulting solutions should be satisfied under every combination of uncertain parameters, including the scenario where every parameter reaches the worst value at the same time. Bertsimas and Sim (2004) introduced a new robustness concept. Instead of requiring the solution to be valid for every possible combination of uncertain parameters, they propose to adjust the level of conservatism of the robust solution in terms of probabilistic bounds of constraint violations. This model assumes that only a subset of the uncertain parameters will change to have a negative effect on the solution, such that the robust solution will be feasible with high probability.

Maggioni, Potra, and Bertocchi (2015) apply Stochastic Programming and Robust Optimization in a transportation problem where the problem is to determine the number of vehicles to replenish cement factories. According to Maggioni et al. the main advantage of Robust Optimization is that it can be solved in polynomial time and has theoretical guarantees on the quality of the solution, which is not the case for Stochastic Programming. Their numerical experiments show that Robust Optimization results in worse objective values due to certitude with relation to satisfying constraints.

A different approach to solving stochastic optimization problems is Sample Average Approximation (SAA) as discussed by Verweij, Ahmed, Kleywegt, Nemhauser, and Shapiro (2003). This Monte Carlo based approach generates a set of random simulation worlds by sampling from a probability distribution for every uncertain parameter. The expected objective function of the stochastic optimization problem is approximated by the average of the samples. We will adopt this technique, since it allows to easily include a wide range of real-world constraints like driving regulations, heterogeneous fleets, vehicle to site compatibilities and many more, while this is not the case for Stochastic Programming and Robust Optimization. Moreover, Sample Average Approximation allows to extend the problem to include other types of uncertainty like uncertain demands. Studying how to solve the DTPP using Robust Optimization is an interesting next research topic.

3

DAIPEX Transportation Planning Problem

The DAIPEX Transportation Planning Problem (DTPP) is an extension of the classical Vehicle Routing Problem (VRP) which is studied extensively in literature. In the DTPP we try to capture a broad array of real-life vehicle routing problems faced by transportation companies. In this chapter we describe the DAIPEX Transportation Planning Problem. We divide the problem definition in four parts: instance data, transportation plan, operational constraints, and the objective. The DTPP is then defined as the problem to find a transportation plan as defined in Section 3.2, that satisfies all operational constraints in Section 3.3 and minimizes the objective function as defined in Section 3.4.

3.1 Instance data

The instance data is comprised of the transportation network, the customer orders and the transportation resources. Together they characterize the problem of a transportation company.

3.1.1 Transportation network

The transportation network contains the customer locations, the central depot and the arcs between them. Let $G = (N_0, A)$ denote the transportation network where $N_0 = \{\mathit{depot}, 1, \ldots, n\}$ is the set of locations and $A = \{(i, j) | i, j \in N, i \neq j\}$ the set of arcs between the locations. The depot is located at location *depot* and $N = N_0 \setminus \{1, \ldots, n\}$ represents a set of customers locations. In general a location represents a single customer, but a location can be shared by multiple customers.

Let $d(i, j) \in \mathbb{R}$ denote the distance of the arc between locations i and j. A distance matrix $D = \{d(i, j) | i, j \in N_0\}$ is defined on A and we assume the arcs have asymmetric distances, that is, traveling from location i to location j does not necessarily have the same distance as traveling from location j to location i. This is the result of one way roads, city centers, and motorways. We assume the transportation network is complete such that for every pair of locations there is an arc with finite distance.

Travel times in the transportation network are subject to random factors. This includes but is not limited to traffic jams, crowded city centers, traffic lights, weather conditions and road works. We include this in our problem definition by assuming that travel times on arcs are continuous random variables. Besides travel times being stochastic, we also see there is a correlation between the time of the day an arc is traversed and the travel time. A clear example is rush hour in the morning and in the evening which causes travel times to be higher compared to traveling just before lunch. We divide the day in time periods such that travel conditions for different departure times within a time period are comparable. Let P be the set of time periods where every time period has its own probability distribution for the travel time. Any random probability distribution can be used in this approach. The travel time on arc (i, j) in time period $p \in P$ is denoted by the continuous random variable $\boldsymbol{T}^p_{i,j}$. This random variable is applicable when the complete travel takes place in time period p. However, it often happens that a travel starts in one time period and continues in one or multiple other time periods. The travel time on arc (i, j) starting at departure time $\boldsymbol{D}$ is denoted by the continuous random variable $\overrightarrow{\boldsymbol{T}}_{i,j}(\boldsymbol{D})$. The relation between $\boldsymbol{T}^p_{i,j}$ and $\overrightarrow{\boldsymbol{T}}_{i,j}(\boldsymbol{D})$ will be explained in Chapter 4. Note that we use bold letters for random variables.

Given a route we are interested in the arrival and departure times at the customer locations. Because the travel times are random variables, arrival and departure times are also random variables. The corresponding probability distributions can be obtained by applying the convolution operator as will be explained in Section 3.3.1.

3.1.2 Orders

Customers have two types of orders: delivery orders and pickup orders. Let O denote the set of orders. For a delivery order a quantity $q_o \in \mathbb{R}$ has to be loaded at the depot and unloaded at a location different than the depot. For a pickup order $o \in O$ a quantity $q_o \in \mathbb{R}$ has to be loaded at a location different than the depot and unloaded at the depot. We define the *stop location* l_o of an order o to be the pickup or delivery location of an order, e.g., for a delivery order it is the location at which the order has to be delivered.

Each order has to be delivered or picked up at its customer within a predefined time window. The time window of the stop location has a release date r_o and a due date d_o such that we express the time window by $[r_o, d_o]$. The release date of a time window is hard and the due date of a time window is soft. That is, when a vehicle arrives before the release date at a customer it is required to wait until the release date starts before the service can start. Arriving after the due date is allowed, but induces a penalty cost as explained in Section 3.4.3. Arriving at the stop location at the due date is considered to be on time, even if the service at the customer does not end before the time window closes. The time window of the depot is denoted by $[r_{depot}, d_{depot}]$.

Each order o has a stochastic service time and depends on the order to be transported. It is denoted by the continuous random variable $\boldsymbol{ST}_o$ and represents the loading or unloading of the order.

We have used the above definition of service times. Another option is to base the service times on stops. If multiple orders have to be serviced at the same location, then the service time not only depends on each separate order but also on the stop (multiple orders for the same location). For example, it takes time before the customer is available to start the service, but when he is available then other orders for this location can be serviced without having to wait again for the customer. We can express this by introducing a stochastic service waiting time for a stop which expresses the waiting time before the service of the first order for this location can start. Next to the stochastic service waiting time there is a stochastic service time for an order which expresses the service time of an order.

Each order o has a service reliability requirement $sr_o \in [0,1]$ expressing that the customer of order o should be serviced on time with a probability of at least sr_o.

3.1.3 Transportation resources

Given a heterogeneous fleet of vehicles V, where vehicle v has capacity Q_v, all orders should be serviced with this set of vehicles. All vehicles are initially located at the depot and they have to finish at the depot at the end of the route. Driving is subject to driving regulations, which is explained in Section 3.3.5.

3.2 Transportation plan

Given instance data as described in Section 3.1, a transportation plan is defined on this instance data and is a set of routes which serve the customers. More formally, a transportation plan is a tuple $\langle R_v, st_v, st_o, std_{v,l} \rangle$, where $R_v = \{o_1, \ldots, o_{|R_v|}\}$ gives the route of vehicle v, st_v gives the start time of vehicle v, st_o gives the start time of the service of order o, and $std_{v,l}$ gives the time when a vehicle starts driving from location l. The DTPP is the problem to find a transportation plan that satisfies all constraints in Section 3.3 and minimizes the objective function as described in Section 3.4.4. In this section we describe what a transportation plan of the DTPP is.

Each element of a route $R_v = \{o_1, \ldots, o_{|R_v|}\}$ corresponds to either the delivery or pickup action of an order. The sequence only contains the actions outside the depot, because the vehicles are preloaded and post unloaded. Preloading a vehicle means loading the orders of the route to be delivered into the vehicle and is done before the start of the route by depot operators. Post unloading a vehicle means that orders which are picked up during the execution of the route are unloaded after arrival at the depot by depot operators. This implies that both loading and unloading times at the depot do not add up to the working time of the driver.

Let $\boldsymbol{A}_o$ be the arrival time at the stop location of order o. Let $\boldsymbol{S}_o$ be the start time of the service of order o. Let $\boldsymbol{D}_o$ be the departure time at the stop location of order o. In Section 3.3 we will explain the relation between these times.

3.2.1 Planning decisions

In Section 3.1 we have explained the instance data of the DAIPEX Transportation Planning Problem. Given the instance data and the transportation plan as defined by tuple $\langle R_v, st_v, st_o, std_{v,l} \rangle$ in the previous section, there are six planning decisions which have to be taken to solve the problem:

1. Which orders are planned on which vehicle?
2. What is the sequence of stops on a route?
3. What is the start time st_v of vehicle v?
4. What is the start time of the service st_o of order o?
5. What is the time $std_{v,l}$ that vehicle v starts driving from location l?
6. When are breaks planned?

The first two planning decisions will be taken by the optimizer and are explained in Chapter 5. We use *optimization algorithm* when we refer to the theoretical algorithm and we use *optimizer* when we refer to the software which implements an optimization algorithm. The third planning decision is taken after fixing the routes and is explained in Chapter 6. The three remaining planning decisions will implicitly be taken, but explicitly considering them can improve the transportation plan.

Deciding on the start time of the service at a customer can allow the driver to take a break before the service. This applies when there is much waiting time in the transportation plan. For example, a driver has to wait 40 minutes before the time window of a customer opens. Because the waiting is less than a break of 45 minutes, the waiting time is normally not counted as a break. Instead of starting the service at the start of the time window, we can decide to have the driver wait five additional minutes such that he can take a break of 45 minutes. This can improve the transportation plan.

The next planning decision that has effect on the solution is the departure time from a location. If a driver has to drive a route which expects to take almost the driving limit, then less driving during rush hour can give a transportation plan which satisfies the driving limit. For example, if a driver starts driving immediately after the service at a customer, he would have to drive an additional 2 hours to the depot during rush hour. He then would have driven 9:05 hours, while the driving limit is 9 hours. If the driver waits 45 minutes at the customer, the rush hour has finished and now driving to the depot only takes 1:45 hours. The driver arrives at the depot after driving 8:50 hours, which is less than the driving limit.

Planning decision 6 is about breaks. We model breaks as an implicit planning decision, because we always plan a break after 4:30 hours. Explicitly modeling breaks as a planning decision can improve a transportation plan, for example when considering time-dependent travel times. Planning a break during rush hour can have as effect that the total driving time of the driver decreases compared to planning a break during off-rush hours.

3.3 Operational constraints

In this section we explain the different constraints that apply to the DAIPEX Transportation Planning Problem. We define a route to be the sequence of orders that are serviced on this route. Together with the fact that each vehicle starts and ends at the depot we get a sequence of arcs that have to be traveled by the vehicle.

3.3.1 Time constraints

Time windows of orders enforce time constraints. Together with travel times we can define the arrival time $\boldsymbol{A}_o$ at the stop location of order o, the start time $\boldsymbol{S}_o$ of the service of order o and the departure time $\boldsymbol{D}_o$ from the stop location of order o. For route $R_v = \{o_1, \ldots, o_{|R_v|}\}$ we define the relation between these times. Let $D_{R_v,depot}$ be the departure time of vehicle v from the depot. The time period p which defines the stochastic travel time is derived from the departure time. Chapter 4 explains how we deal with travel times longer than the size of a time period.

The arrival time at the stop location of the first order o_1 of route R_v is defined as follows:

$$\boldsymbol{A}_{o_1} = D_{R_v,depot} + \overrightarrow{\boldsymbol{T}}^{p}_{depot,l_{o_1}}$$

where time period p corresponds to the time period in which the departure time from the depot $D_{R_v,depot}$ falls and will be explained in Chapter 4.

The service start time of order o_i is defined by the maximum of the arrival time and the release date of the order:

$$\boldsymbol{S}_{o_i} = \max\{\boldsymbol{A}_{o_i}, r_o\}$$

The departure time from the stop location of order o is defined by the convolution of the stochastic service time and the start time of the service:

$$\boldsymbol{D}_o = \boldsymbol{S}_o * \boldsymbol{ST}_o$$

Arrival at the stop location of order o_{i+1} is the convolution of the departure time from the previous location (stop location of order o_i) and the travel time between these two locations. Because of time-dependent travel times, the travel time between the two locations depends on the departure time.

$$\boldsymbol{A}_{o_{i+1}} = \boldsymbol{D}_{o_i} * \overrightarrow{\boldsymbol{T}}_{l_{o_i}, l_{o_{i+1}}}(\boldsymbol{D}_{o_i})$$

The arrival time at the depot is found in a similar way:

$$\boldsymbol{A}_{depot} = \boldsymbol{D}_{l_{|R_v|}} * \overrightarrow{\boldsymbol{T}}_{l_{|R_v|}, depot}(\boldsymbol{D}_{l_{|R_v|}})$$

The convolutions involving travel time possibly involve multiple time periods.

3.3.2 Service reliability requirement

Each customer should be serviced with the service reliability requirement of its order. The probability of arriving at the stop location of order o before the due date d_o should be at least sr_o:

$$\mathbb{P}(\boldsymbol{S_o} \leq d_o) \geq sr_o$$

Violating this constraint incurs constraint costs as explained in Section 3.4.3, which means that violating the service reliability requirement can be advantageous compared to solving it.

3.3.3 Vehicle constraints

We assume that each order can be transported by every vehicle in the fleet. An order has to be transported by exactly one vehicle, that is, an order cannot be split up and transported by more than one vehicle:

$$\left|\{v \in V : o \in R_v\}\right| = 1 \qquad o \in O$$

3.3.4 Capacity constraints

Vehicle v has a capacity of Q_v which should not be exceeded at any moment during the execution of route R_v. Let $L_{v,depot}$ be the loaded quantity after preloading vehicle v. We introduce γ_o which indicates whether order o is a pickup order ($\gamma_o = 1$) or a delivery order ($\gamma_o = -1$). The capacity of vehicle v should not be exceeded at any time:

$$L_{v,depot} + \sum_{i=1}^{k} \gamma_{o_i} \cdot q_{o_i} \leq Q_v \qquad \forall k,\ 0 \leq k \leq |R_v|$$

3.3.5 Legal constraints

Vehicles are driven by drivers who must adhere to driving regulations. We use the European driving regulations as defined in Regulation (EC) No. 561/2006 by the European Union (2006). We distinguish driving time and working time. The driving time is the actual time the driver controls his vehicle. The working time is the time between two daily rests and includes activities like driving, loading and unloading, but also cleaning the vehicle or doing administrative work.

The driver is allowed to drive at most 4:30 hours without taking a break. A break is either first a sub break of 15 minutes and a second sub break of 30 minutes or a full break of 45 minutes. We will use the breaks of 45 minutes in this problem, where a break is planned after driving exactly 4:30 hours. This means that driving 4:29 hours can be done without taking a break, but when driving 4:31 hours a break is planned after 4:30 hours with a remaining driving time of 1 minute. The reason for this is that it is not allowed to drive more than 4:30 hours without taking a break and can result in a high penalty. A driver is allowed to take a break while waiting for the release time of an order in case the waiting time is at least the break period of 45 minutes.

Besides a limit on consecutive driving time, we also consider the daily driving limit and the daily working limit. A driver is allowed to drive 9 hours on one day. Similar to the service reliability requirement we have a driving limit reliability requirement: we require that the probability that the driving time for route $R = \{o_1, \dots, o_{|R_v|}\}$ is at most 9 hours should be at least dr_R:

$$\mathbb{P}\left(\boldsymbol{T}_{depot,l_{o_1}} + \boldsymbol{T}_{l_{o_{|R_v|}},depot} + \sum_{1 \leq i < |R|} \boldsymbol{T}_{l_{o_i},l_{o_{i+1}}}(D) \leq 9\right) \geq dr_R \tag{3.1}$$

Section 3.4.3 explains the constraint costs incurred when violating this constraint, which can be beneficial compared to solving it. Moreover, the regulations state that a driver is entitled to have a daily rest of at least 11 hours every day in which the driver is not performing any actions related to his work. This means the daily working limit is 13 hours. Besides a normal daily rest there is also a reduced daily rest which is only allowed three times during two weeks. The reduced daily rest is 9 hours which means the working limit on those days is 15 hours. We will not go into detail on the regulations related to driving multiple days.

3.4 Objective

The objective of the DAIPEX Transportation Planning Problem is to minimize the costs and constraint costs of a transportation plan. The cost has different components which are explained in this section. We distinguish vehicle-dependent costs, driver-dependent costs and constraint costs.

Let d_{R_v} denote the distance traveled in route R_v:

$$d_{R_v} = d(depot, l_{o_1}) + \sum_{i=1}^{|R_v|-1} d(l_{o_i}, l_{o_{i+1}}) + d(l_{o_{|R_v|}}, depot) \tag{3.2}$$

Table 3.1 shows five different cost factors used in the definition of the objective.

c_{vu}	cost per vehicle usage
c_{vk}	cost per driven kilometer for a vehicle
c_{vh}	cost per hour of vehicle usage
c_{du}	cost per driver usage
c_{dh}	cost per worked hour of drivers

Table 3.1: Different cost factors

3.4.1 Vehicle-dependent costs

Using a vehicle in a transportation plan induces three types of costs: fixed costs, costs per driven kilometer and costs per hour of resource usage as can be seen in Table 3.1. When using a vehicle, we firstly have to pay a fixed amount c_{vu}. Next to this we pay an amount of c_{vk} for every driven kilometer. Furthermore, we pay an amount of c_{vh} for every hour that the vehicle is used. Together with Formula 3.2 and the expected value of the driving duration (stochastic arrival time minus deterministic departure time from depot) we calculate the vehicle-dependent costs for route R_v:

$$CostsVehicle_{R_v} = c_{vu} + c_{vk} \cdot d_{R_v} + c_{vh} \cdot \mathbb{E}(\boldsymbol{A}_{depot} - D_{R_v,depot}) \tag{3.3}$$

3.4.2 Driver-dependent costs

Next to vehicle-dependent costs, these vehicles also have to be driven by a driver who has to be paid as well. Assigning a driver to a route induces two types of costs: fixed costs and costs per worked hour of a driver as can be seen in Table 3.1. Planning in a driver costs a fixed amount of c_{du}. For every worked hour of a driver we have to pay c_{dh}. This gives the driver-dependent costs for route R_v:

$$CostsDriver_{R_v} = c_{du} + c_{dh} \cdot \mathbb{E}(\boldsymbol{A}_{depot} - D_{R_v,depot}) \tag{3.4}$$

3.4.3 Constraint costs

Each order should be served with a required service reliability requirement. However, there are problem instances in which it is not possible to meet the service reliability requirement of all orders or we want a trade-off between satisfying a constraint with high constraint and not satisfying a constraint with a large reduction in costs. The same holds for the driving limit reliability constraint and the capacity constraint.

We define the *Virtual Stochastic Costs* as the costs for the vehicle and the driver; and the constraint costs for not meeting all constraints such that we can compare transportation plans even if violated constraints.

3.4.3.1 Service reliability requirement

Let o be an order which has a service reliability requirement of sr_o and a realization of the service reliability requirement of rsr_o meaning that with a probability of rsr_o the vehicle arrives before the time window of the stop location closes. We define the violation for o: $violation_o = \max(sr_o - rsr_o, 0)$. The service costs are positive if there is a violation of the service reliability requirement:

$$ServiceCosts_o = \begin{cases} 0 & \text{if } violation_o = 0 \\ factor_s \cdot (violation_o + offset_s)^{power_s} & \text{if } violation_o > 0 \end{cases} \tag{3.5}$$

where $factor_s, offset_s$ and $power_s$ are equal factors for all orders. If an order is served with at least the service reliability requirement then the service costs are zero for this order.

We define the service costs for a route as the sum of service costs of its orders:

$$TotalServiceCosts_{R_v} = \sum_{i=1}^{|R_v|} ServiceCosts_{o_i} \tag{3.6}$$

3.4.3.2 Driving limit reliability requirement

Violating the driving limit is not allowed according to European Regulations. To solve this problem in practice different alternatives are available which are discussed in Section 7.1. We will use the constraint as expressed by Formula 3.1 for the driving limit.

Let R be a route which has a driving limit reliability requirement of dr_R and a realization of the driving limit reliability requirement of rdr_R meaning that with a probability of rdr_R the driving time for route R is at most 9 hours. We define the violation for $R : violation_R = \max(dr_R - rdr_R, 0)$. The driving limit costs are positive if there is a violation of the driving limit reliability requirement:

$$DrivingLimitCosts_R = \begin{cases} 0 & \text{if } violation_R = 0 \\ f_{dr} \cdot (violation_R + vo_{dr})^{p_{dr}} & \text{if } violation_R > 0 \end{cases} \tag{3.7}$$

where f_{dr}, vo_{dr} and p_{dr} are equal factors for all routes. If the probability that the driving time of a route is below the driving limit with at least the driving limit reliability requirement, then the driving limit costs for this route are zero.

3.4.3.3 Capacity constraint

Each transportation resource R_v has a maximum capacity Q_v. Violating this incurs a constraint costs. We define the constraint violation on order level: for every pickup or delivery we verify if the maximum capacity is exceeded. We use the same terminology as in 3.4.3: γ_o indicates whether order o is a pickup order ($\gamma_o = 1$) or a delivery order ($\gamma_o = -1$) The violation for order j on resource R_v is defined as follows:

$$violation_{R_v,j} = \max\left(L_{R_v,depot} + \sum_{i=0,\ i\leq j}^{|R_v|} \gamma_{o_i} \cdot q_{o_i} - Q_{R_v}, 0 \right) \tag{3.8}$$

The capacity constraint costs for order o on resource R_v are positive if there is a violation of the capacity constraint:

$$CapacityCosts_{R_v,j} = \begin{cases} 0 & \text{if } violation_{R_v,j} = 0 \\ f_{ca} \cdot (violation_{R_v,j} + vo_{ca})^{p_{ca}} & \text{if } violation_{R_v,j} > 0 \end{cases} \tag{3.9}$$

where f_{ca}, vo_{ca} and p_{ca} are equal factors for all routes.

We define the capacity constraint costs for a route as the sum of the capacity constraint costs of its orders:

$$TotalCapacityCosts_{R_v} = \sum_{i=1}^{|R_v|} CapacityCosts_{R_v,o_i} \tag{3.10}$$

3.4.4 Virtual Stochastic Costs

The total objective is a combination of the vehicle-dependent costs, driver-dependent costs and violation costs. We can use the costs in Formulas 3.3, 3.4 and 3.6 to calculate the total objective named *Virtual Stochastic Costs*:

$$\begin{aligned} Virtual\ Stochastic\ Costs = \sum_{v\in V} (&CostsVehicle_{R_v} + CostsDriver_{R_v} + \\ &TotalServiceCosts_{R_v} + \\ &DrivingLimitCosts_{R_v} + \\ &TotalCapacityCosts_{R_v}) \end{aligned} \tag{3.11}$$

4

Stochastic travel times

The optimizer developed in this research for the DAIPEX Transportation Planning Problem is tested on real-life instances for which travel time data is used. In Chapter 3 we assumed we have a function which provides the travel time between any two locations. In Section 4.1 we explain how we constructed this travel time function. Section 4.2 explains how we obtain travel time distribution for every arc where that distribution is originally missing. This technique is useful when not enough data is available, for example in case one is frequently faced with new locations, which is a common use case in areas of logistics. This is especially useful when one gets travel time distributions from logs coming from handheld or on board devices. Such logs are mostly gathered through a Transportation Management System (TMS) and provide travel time data for a company's own trucks. In this thesis we used this data completion approach to obtain missing travel time distributions as TomTom was not yet able to provide us with all distributions in the available time.

4.1 Travel time representation

Let the continuous random variable $\boldsymbol{T}^p_{i,j}$ denote the travel time on arc (i, j) in time period $p \in P$ as introduced in Chapter 3. This random variable is characterized by the quantile function $Q^p_{i,j}(q)$, where q is a cumulative probability between 0 and 1. Given a quantile q, with $0 \leq q \leq 1$, the quantile function expresses that the travel time from location i to j in time period p is $Q^p_{i,j}(q)$. Figure 4.4 shows the quantile function for the travel time from Zaandam to Utrecht. We have a set of time periods P and every time period $p \in P$ has a start and end time: p_{start} and p_{end} denoted as $p = [p_{start}, p_{end}]$.

4.1.1 Traversing time periods

This section explains how to calculate the travel time when traversing a link using multiple time periods: it shows the relation between $\boldsymbol{T}^p_{i,j}$ and $\overrightarrow{\boldsymbol{T}}_{i,j}(\boldsymbol{D})$, where i and j

are locations, p is a time period and D is a departure time. The first calculation does not take into account breaks. The next step is to include breaks in the algorithm according to the driving regulations explained in Section 3.3.5. We will see that the calculation of breaks should be intertwined with the travel time calculation.

4.1.1.1 Excluding breaks

When calculating with time-dependent travel times, we have to be aware that a vehicle which travels along a link can cross boundaries of time periods. Calculating the travel time naively causes the travel times to violate the ‘first-in-first-out‘ property (FIFO). This property states that if vehicle 1 departs before vehicle 2 from location i, then vehicle 1 should arrive at location j before vehicle 2.

Consider the following example with two time periods p_1 and p_2 with $p_1 = [0, 5]$ and $p_2 = [5, 10]$. Vehicle 1 departs from location i to location j at time 4 and vehicle 2 departs from location i to location j at time 5. Let $Q^{p_1}_{i,j}(q) = 4$ and $Q^{p_2}_{i,j}(q) = 2$. Using these travel times we get that vehicle 1 arrives at location j at time $4+4 = 8$. Vehicle 2 arrives at location j at time $5 + 2 = 7$. In this example it seems beneficial for vehicle 1 to wait until the start of the second time period such that it arrives earlier at location j, while in reality this should not be possible.

The example shows that we should take the FIFO property into account when calculating time-dependent travel times. Ichoua, Gendreau, and Potvin (2003) introduce a model based on time-dependent travel speeds. Instead of assuming the travel time function on a link is a step function of the time of the day, Ichoua et al. assume the travel speed is a step function of the time of the day. The result is that the travel time is a piecewise continuous function over time. The travel speed makes ‘jumps‘ at the boundaries of the time periods, which could be solved by assuming the acceleration is a step function of the time such that the travel speed is a piecewise continuous function over time. This would give a quadratic function for the travel time.

We will use the model proposed by Ichoua et al. to calculate time-dependent travel times. The idea is to calculate how much of the complete travel is in a time period. If the complete travel can be finished before the time period ends the algorithm terminates. If only a part of the travel can be finished before the time period ends, we calculate which part is traversed in the time period and which part is not yet traveled. In the latter case we continue the algorithm with the next time period and a reduced part of the link which should be traveled. Algorithm 1 shows how to compute the travel time.

Let’s recall the example from the beginning of this section. We have two time periods p_1 and p_2 with $p_1 = [0, 5]$ and $p_2 = [5, 10]$. Vehicle 1 departs from location i to location j at time 4 and let $Q^{p_1}_{i,j}(q) = 4$ and $Q^{p_2}_{i,j}(q) = 2$. Using the algorithm we find that vehicle 1 can traverse $\frac{5-4}{4} = 0.25$ of the link in the first time period. This means that in the next time periods vehicle 1 has to traverse 0.75 of the link. In the second time period $Q^{p_2}_{i,j}(q) = 2$, so vehicle 1 traverses $2 \cdot 0.75 = 1.5$ in the

Algorithm 1: Calculation time-dependent travel time excluding breaks.

```
input  : locations $a$ and $b$, departure time $d$, cumulative probability $q$
output: time-dependent travel time between $a$ and $b$ when departing from $a$
        at time $d$ with cumulative probability $q$
1  begin
2      $\phi \leftarrow 1$
3      $T \leftarrow 0$
4      while $\phi > 0$ do
5          $period \leftarrow$ time period $p \in P$ with $p_{start} \leq d < p_{end}$
6          $\delta \leftarrow \min\left(\phi, \frac{p_{end} - d}{Q^p_{i,j}(q)}\right)$        // ratio traversed in time period
7          $T \leftarrow T + \delta \cdot Q^p_{i,j}(q)$
8          $\phi \leftarrow \phi - \delta$
9          $d \leftarrow p_{end}$        // departure time for next time period
10     end
11     Return $T$
12 end
```

second time period which is possible because time period p_2 is 5 hours. Vehicle 1 has finished traversing the link which means the algorithm has finished. We get an arrival time at location j of $4 + 1 + 1.5 = 6.5$. We now see that it is no longer beneficial for vehicle 1 to wait until the start of the second time period.

4.1.1.2 Including breaks

In Section 3.3.5 we have seen that a driver is allowed to drive 4:30 hours consecutively and must then take a breaks of 45 minutes. When using a single time period we can afford to first calculate the travel time for an arc and afterwards calculate whether a break should have been taking during the travel and simply adding it to the travel time. When using time-dependent travel times it is of the uttermost importance to calculate the start time of breaks while calculating the travel time.

We will show this using an example with two consecutive time periods p_1 and p_2 with $p_1 = [0, 5]$ and $p_2 = [5, 10]$. A driver is allowed to drive 4.5 and a break takes 0.75. Let $Q^{p_1}_{i,j} = 5$ and $Q^{p_2}_{i,j} = 2$. A vehicle leaves location i at time 4.25 to go to location j. Consider the difference between planning a break during a traffic jam (time period p_1) or planning a break during quite hours (time period p_2). In the former case let's plan the break directly at the start of the travel at 4.25. The break finishes at $4.25 + 0.75 = 5$. This coincides with the end of the first time period and thus with the start of the second time period. The travel can completely be done in this second time period which means vehicle 1 arrives at location j at time $5 + 2 = 7$.

In the latter case the vehicle drives 0.75 in first time period which is $\frac{0.75}{5} = 0.15$

of the complete travel. This means the vehicle has to travel $(1 - 0.15) \cdot 2 = 1.7$ in the second time period, besides the break of 0.75. This gives an arrival time at location j of $4.25 + 0.75 + 1.7 + 0.75 = 7.45$. We see that the time when a break is planned can have effect on the travel time for time-dependent travel times. It can be beneficial to plan a break during a traffic jam, since the vehicle cannot cover much distance during this period. Therefore, to include breaks in Algorithm 1, we have to calculate the exact time when breaks occur. We also have to consider the cases for breaks being longer than a time period or planning multiple breaks in a single time period.

Algorithm 2 shows how to compute the time-dependent travel time between locations including breaks. In the first 6 lines of Algorithm 2 various parameters are initialized. The idea of the algorithm is equal to Algorithm 1: we calculate how much of the travel has been done in a time period and which part is left.

When calculating time-dependent travel times with breaks, we have to take into account how long the driver has driven before this travel has started (this should also be done when using a single time-period, but can be done afterwards in this case). Therefore, the driving time since the last break T_{before} is an argument of the algorithm. Next to the driving time since the last break we also maintain the break duration we should take in the next period, because the last period has ended. If the break to take does not fit in the time period (lines 9 to 12), the period is completely planned as a break and the break that is left is passed to the next time period after updating the counters for the breaks and the departure time.

If the break fits in the period, the break is completely planned (line 14). We calculate the travel time which is possible in the time period without taking into account breaks and which has not been traveled. If no break is required during this duration (travel time before period plus travel time in period is less than maximum driving time between breaks), the driving can start and we update the relevant counters (lines 33 and 34). If a break is required during the driving, we start the driving until the break is required. The break might not fit in the period, so the break is partially done and the rest is forwarded to the next time period (lines 18 to 21).

If the break fits in the time period, the break is done. We are now in the situation that driving can start and can be alternated by breaks. Lines 23 to 30 show that driving and breaks alternate until either the time period has ended or the driving in this time period has ended.

While executing the algorithm we should maintain the travel time T, the break in the travel $breakInTravel$ and the driving time since the last break $drivingAfterLastBreak$. They will be returned at the end. We can see that calculating time-dependent travel times including breaks is more complicated than 1) time-dependent travel time without breaks and than 2) travel times including for a single time period.

Algorithm 2: Calculation time-dependent travel time including breaks.

```
input : locations $a$ and $b$, departure time $d$, cumulative probability $q$,
        driving time since last break $T_{before}$
output: time-dependent travel time between $a$ and $b$ when departing from $a$
        at time $d$ with cumulative probability $q$, break duration $B_{taken}$ taken
        during the travel, travel time $T_{afterBreak}$ after the last break
begin
    $\phi \leftarrow 1$   // ratio left to travel
    $T \leftarrow 0$   // total travel time
    $breakInTravel \leftarrow 0$   // total break duration during travel
    $drivingAfterLastBreak \leftarrow 0$   // driving duration since last break
    $breakToTakeInPeriod \leftarrow 0$   // required break duration for next period
    while $\phi > 0$ do
        $period \leftarrow$ time period $p \in P$ with $p_{start} \leq d < p_{end}$
        if breakToTakeInPeriod does not fit in period then
            Do part of break
            Update counters for breaks and departure time $d$
            Continue with next period
        else
            Do break   // break fits in period
            $driving \leftarrow$ travel time in period using $\phi$ and $p_{end}$ without breaks
            if break needed during driving then
                Start driving until break required
                if break does not fit in period then
                    Do part of break
                    Update counters for breaks and departure time $d$
                    Continue with next time period
                else
                    Do break   // break fits in period
                    $\phi_2 \leftarrow 0$   // Large period requires multiple breaks
                    while $\phi_2 > 0$ and period has not ended do
                        Driving until break required or end period
                        Do break
                        Update counters for driving, break and departure time $d$
                        Continue with next period
                    end
                end
            else
                Do driving
                Update counters for driving, breaks and departure time $d$
            end
        end
    end
    Return $T, breakInTravel, drivingAfterLastBreak$
end
```

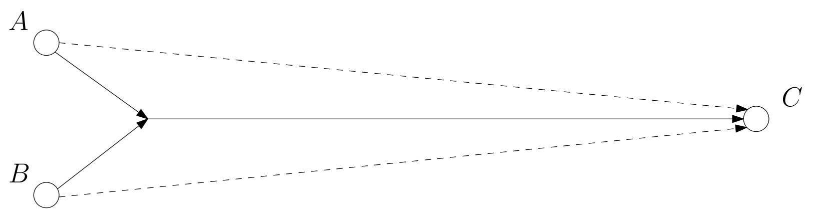


Figure 4.1: Arcs (A, C) and (B, C) have much of the route in common, while independent sampling does not take this into account.

4.1.2 Simulations

In Section 2.4 we have discussed various solution approaches for optimization under uncertainty. As explained, we will use the Sample Average Approximation technique as described by Verweij et al. (2003) to solve the DTPP. Therefore, instead of calculating with the probability distributions of the travel times in our algorithms directly, we will create a set of simulation worlds based on the probability distributions. By sampling from the probability distributions the realized travel times in the simulation worlds approximate the travel time distributions.

Let S be the set of simulation worlds we use. For every arc (i, j), where i and j are locations from N_0, and for every simulation world $s \in S$ we need a sample $q^s_{i,j}$ between 0 and 1 to get the realized travel time from the travel time distribution. The sample $q^s_{i,j}$ expresses the traffic flow on the arc. A sample close to 1 has a meaning of high traffic flow: traveling this arc takes longer than on average. We discuss three methods to generate samples for an arc: independent sampling, sampling based on maximum distance clustering and sampling based on hierarchical clustering.

4.1.2.1 Independent sampling

Using independent sampling we generate a random sample $q^s_{i,j}$ from $X \sim U(0, 1)$ for every arc for every simulation. This gives $|N_0|^2$ samples for every simulation world, where N_0 is the set of locations. This method has the disadvantage that there is no dependence between the travel times, despite in reality travel times are correlated. For example, when traveling from location A to C and from B to C where locations A and B are close and C is further away. When traveling on those two arcs they have much of the route in common shown by the middle arc in Figure 4.1. This means that when two vehicles depart from location A and B at the same time to go to location C, they suffer from the same traffic flow on a major part of their route. Using independent sampling we assume the travel times are independent shown by the dashed arcs in Figure 4.1. The triangle inequality is not satisfied in this case. The travel time from A to C can be longer than driving from A to B and then to C. To overcome this, we introduced clustering before sampling.

4.1.2.2 Sampling based on maximum distance clustering

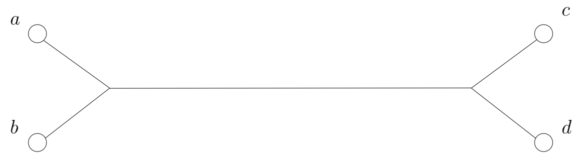
To overcome the problem we saw with independent sampling, we cluster the locations and we assume the traffic flow between the locations of one cluster and the locations of another cluster are equal. We show two algorithms to cluster the locations, where the second algorithm is discussed in the next section. The first algorithm places locations in the same cluster if they are within a maximum distance d of each other. The idea behind this is shown in Figure 4.1 and discussed in the previous paragraph: if two arcs share a major part of their route, then they should have the same traffic flow. Assume the N_0 locations are clustered in a set of clusters C. Instead of generating a sample for every arc using the set of locations N_0, we generate a sample for every arc using the set of clusters C resulting in $|C|^2$ samples for every simulation world. This means that we use the same sample for two arcs (i, j) and (k, l) if locations i and k are in the same cluster and locations j and l are in the same cluster.

Pickup and delivery locations are often not uniformly distributed over the area, but instead many locations are close to each other in cities while outside of the cities the locations are further apart. Using a fixed maximum distance has as result that there are large differences in the sizes of the clusters. For example, we use a maximum distance of 4 km such that all addresses which are within 4 km of each other are in the same cluster. Clustering 318 customer locations from a transportation company with a maximum distance of 4 km gives 100 clusters where the five largest clusters together contain 47% of the addresses but most of the clusters contain only 1 or 2 addresses.

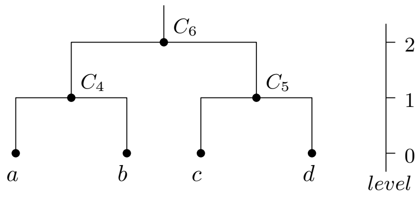
We generate two samples for every pair of clusters (arcs are directed) resulting in many samples between small clusters. We then see a similar effect as independent sampling: when a major part of the route is equal for two arcs the traffic flow can still be very different. On the other hand if we take a larger maximum distance then the largest cluster becomes even larger, while we still have many small clusters. Another disadvantage of this approach is that a cluster can consist of two smaller clusters but they are both within the maximum allowed distance from each other such that the two smaller clusters get merged into the large cluster. In this case we assume all arcs in the largest cluster have the same traffic flow, but because the locations cover a large area we do not believe sampling using this method gives the best results.

4.1.2.3 Sampling based on hierarchical clustering

The third method to generate samples we have explored is based on hierarchical clustering. The idea is that pairs of arcs which have a large part of their route in common should have the same traffic flow. We use an agglomerative hierarchical clustering algorithm where we start with each location in its own cluster and pairs of clusters are merged when moving up the hierarchy. For the distance between two locations we use the *geodesic* distance (shortest path between two points on the



(a) Arcs (a,c), (a,d), (b,c) and (b,d) have much of their route in common.



(b) Dendrogram of hierarchical clustering.

Figure 4.2: Hierarchical clustering

Earth's surface). For the distance between two clusters A and B we use the average linkage clustering:

$$\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} geodesic(a,b) \tag{4.1}$$

Figure 4.2a shows four locations a, b, c and d where a and b are close to each other and c and d are close to each other. The hierarchical clustering of these four locations is shown in Figure 4.2b, where a cluster is shown using a circle. At the lowest level every location is in its own cluster. In the second level locations a and b are in one cluster and locations c and d are in another cluster. Moving to the highest level gives a single cluster with all locations.

For every simulation world $s \in S$, we generate a sample $q^s_{i,j}$ with $q^s_{i,j} \sim U(0,1)$ for every cluster in the hierarchical clustering. We define an ancestor of a cluster C as a cluster which has all locations from cluster C and which has a higher level. For the arc from location i to j in simulation world s, we use the sample $q^s_{i,j}$ from the first common ancestor cluster of clusters C_i and C_j. For example in Figure 4.2 when we want a sample for the arc from location A to location C we find the first common ancestor cluster to be cluster C_6 at level 2. For the simulation world s we use the sample $q^s_{a,c}$ of cluster C_6 to instantiate the travel time for arc (a, c).

Figure 4.3 shows a part of the hierarchical clustering of 319 locations where 3 locations a, b, c are highlighted with green squares and clusters are shown with their convex hull. We want to generate samples for arc (a, b) and (a, c). For arc (a, b) we find the first common ancestor cluster of locations a and b to be the cluster shown in Figure 4.3 with the dark blue convex hull. By visual inspection we see that for arc (a, c) the same cluster is the first common ancestor of locations a and c. The result is that the same sample $q, 0 \leq q \leq 1$, is used for the two arcs. This means that that when driving from a to b we expect an equal traffic flow as when driving from a to c.

This sampling approach can be extended to take into account the direction of an arc, which is the case for the sampling based on maximum distance clustering. We generate two samples for every cluster in the hierarchy. The first sample for arcs with a from-location in the left child and with the to-location in the right child and the other sample for the other direction (in Figure 4.2b the left child of cluster C_6 is cluster C_4 and the right child is cluster C_5).

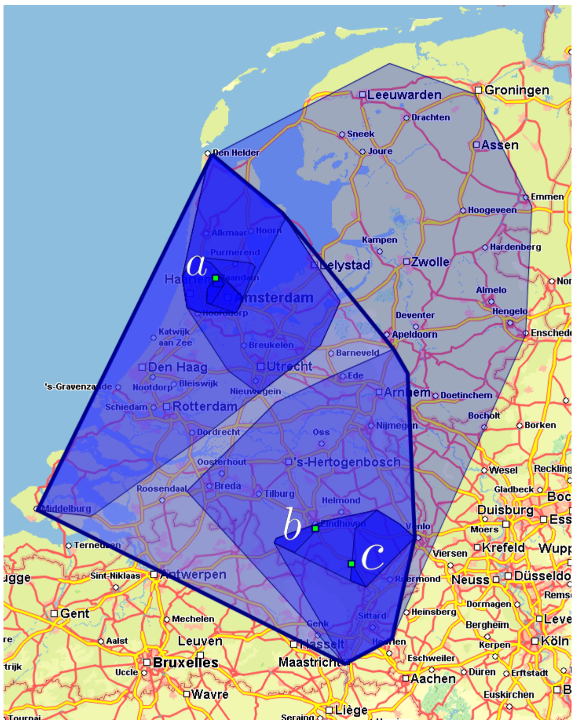


Figure 4.3: Finding the common ancestor of clusters for sampling based on hierarchical clustering.

4.1.2.4 Overview

In this section we have seen three methods methods to generate samples for an arc: independent sampling (4.1.2.1), sampling based on maximum distance clustering (4.1.2.2) and sampling based on hierarchical clustering (4.1.2.3). When using independent sampling we see many violations of the triangle inequality in the simulation worlds. This is undesirable as it does not accurately represent the real world. We proposed two solutions for this issue. Using sampling based on maximum distance clustering gives a couple of large clusters while many clusters are small, because the locations are not uniformly spread over the total area. To overcome this, we have proposed a second solution using sampling based on hierarchical clustering. No maximum distance is needed for this approach. The advantage is that similar arcs (arcs with the from-locations close to each other and the to-locations close to each other) use the same sample. This applies for arcs with a large travel time, but also for arcs with a short travel time.

4.1.3 Service reliability

Given the set of simulation worlds S and an order o, we approximate the service reliability sr_o and whether this meets the service requirement s_o for order o. In Section 3.2 we defined $\boldsymbol{S}_o$ as random variable representing the start time of the service of order o. Let S_o^s be the start time of the service of order o in simulation world $s \in S$. We define the function $OnTime(s, o)$ to express if the service of order

o starts on time in simulation world $s \in S$:

$$OnTime(s, o) = \begin{cases} 1 & \text{if } S_o^s \leq d_o \\ 0 & \text{else} \end{cases} \tag{4.2}$$

For the approximation of the service reliability sr_o we calculate in which percentage of the simulation worlds the service of order o starts before the order due date:

$$sr_o = \frac{1}{|S|} \sum_{s \in S} OnTime(s, o) \tag{4.3}$$

4.2 Travel time data

The IE&IS group of the TU/e has developed data aggregation algorithms which allow to obtain probability distributions for travel times from TomTom. They make use from an API provided by TomTom (*TomTom Custom Travel Times API* (n.d.)) and process the data were needed. In Section 4.2.1 we explain which format we defined to communicate requests and responses between Quintiq and the IE&IS group of the TU/e. We have a limited set of locations for which we have the full submatrix of travel time distributions. Section 4.2.2 explains how we approximate the travel time distributions for missing arcs based on the available probability distributions. In this section we will explain how the travel time data looks like and how we will use it when applying sampling average approximation.

Using historical traffic data from TomTom we get a probability distribution representing the travel time on arc (i, j) in time period p. Next to the average travel time of an arc, TomTom provides us with incomplete probability distributions: the travel times for the cumulative probabilities in the set $\{0.05, 0.1, 0.15, \ldots, 0.95\}$ are provided. The cumulative probabilities define a piecewise-linear quantile function between two locations on the interval $[0.05, 0.95]$. The black circles in Figure 4.4 show the received quantile function for the travel time from Zaandam to Utrecht.

We are missing travel times for the start and trail of the distributions: $[0, 0.05]$ and $[0.95, 1]$. These values are required when generating the simulation worlds: if random sample q is in the interval $[0.95, 1]$ we need to know the travel time at the 1.0 quantile to interpolate the travel time for sample q. We solve this problem by computing the cumulative probabilities 0 and 1, such that the expected value of the distribution is equal to the average provided by TomTom. They have provided us with n data points $(q_i, t_i) \in Q$ with cumulative probability q_i and maximum travel time t_i, where $q_1 = 0.05$ and $q_n = 0.95$. Since Q is a cumulative probability function, we know that if $q_i < q_{i+1}$, then $t_i \leq t_{i+1}$. The expected value of the piecewise-linear quantile function can be calculated using the probability mass of every interval. We use the average travel time of an interval for the time representing the bin. The expected value of the probability distribution can be calculated using the follow formula:

$$E(Q) = \frac{1}{2} \sum_{i=1}^{n-1} (q_{i+1} - q_i)(t_i + t_{i+1}) \tag{4.4}$$

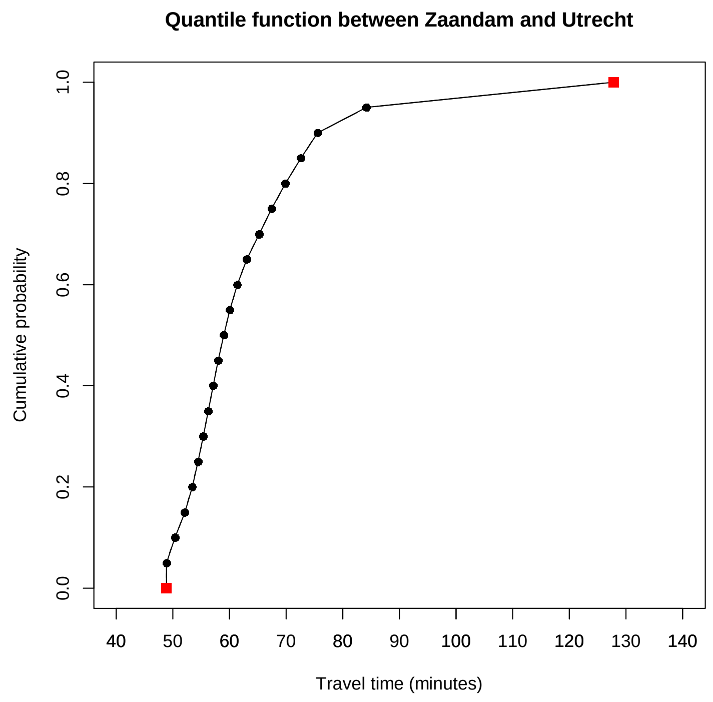


Figure 4.4: Quantile function for travel times between Zaandam and Utrecht including calculated quantiles (shown by red squares).

In the data provided by TomTom we have $q_{i+1} - q_i = 0.05$ for every $1 \leq i < n$. Lekkerkerker (2016) uses this formula to calculate the travel time for the last quantile q_{n+1}, such that the average of the received distribution is equal to the expected value of the cumulative probability function:

$$t_{n+1} = \frac{2\mu - \sum_{i=1}^{n-1}(q_{i+1} - q_i)(t_i + t_{i+1})}{1 - q_n} - t_n \tag{4.5}$$

where μ is the average of the received cumulative probability function. To find t_{n+1} they set $t_0 = t_1$ to get the function with the lowest variance.

Figure 4.4 shows the complete quantile function for the travel times between Zaandam and Utrecht. The computed travel times at quantiles 0 and 1 are shows with red squares. We use linear interpolation between known quantiles to get travel times for other quantiles.

4.2.1 Communication Quintiq and TU/e

The probability distributions for the travel times, also known as travel time distributions, used in this research come from the IE&IS group of Eindhoven University of Technology (TU/e) hereafter referred to as TU/e. Quintiq can request a set of travel time distributions between addresses. We have defined a format to ease this process. The format can and should be used in general to communicate travel time distributions, this as part of the data that is required to define the DAIPEX Transportation Planning Problem. We explain the format of the request and response. Appendix A shows use cases of how this format can be used. The format is split in a request and a response.

4.2.1.1 Request

The general idea is to define groups of addresses and to request travel times between these groups of addresses. This should give a submatrix of travel time distributions.

The request file is structured as follows: on each line an address (identifier, latitude and longitude), then a white line, then on each line a group, then a white line and then on each line a group combination from which the travel time distributions are requested. The travel time distributions are requested from each of the addresses in the first group to each of the addresses in the second group.

The specification is shown below:

```
--- Start of file ---
ID address 1, latitude, longitude
ID address 2, latitude, longitude
...
ID address n, latitude, longitude

GROUP1, ID address 1, ID address 2
GROUP2, ID address 3, ID address n

GROUP1, GROUP2
--- End of file ---
```

The coordinates are expressed in geographic coordinates in decimal degree notation (latitude bounded by $\pm$ 90° and longitude bounded by $\pm$ 180°). Both latitude and longitude are formatted using a point (.) as decimal separator. An example latitude is 52.471592.

4.2.1.2 Response

The response contains for each requested address pair the travel time distributions. The request does not contain the time period, but after a study of the IE&IS group we have decided to use the following periods:

- 0:00 - 7:15
- 7:15 - 9:30
- 9:30 - 17:00

- 17:00 - 19:00
- 19:00 - 23:59

A request for the travel time distribution from address A to address B should a response of 5 travel time distributions: one for each period of the day.

The response file starts with one line of headers. This will be:

```
Origin,Destination,Day,Time,Duration,Average,Median,[Q1],...,[Qn]
```

with [Qi] replaced by a value between 0 and 1 that represent the quantile.

Then every line below these headers will be data, according to the specified headers. So a line looks like:

```
[Origin],[Destination],[Day],[Time],[Duration],[Average],
[Median],[[Q1]],[[Q2]],...,[[Qn]]
```

where we replace:

Origin: The identifier of the origin location. This is a String.

Destination: The identifier of the destination location. This is a String.

Day: The day of the week. The days are integer values between 1 and 7, with 1 = Monday, 2 = Tuesday, ..., 7 = Sunday. When travel time distributions are aggregated over work days, WorkDays is also a valid value for [Day].

Time: The start time of the time window.

Duration: The duration of the time window.

Average: The average travel time.

Median: The median travel time.

[Qi]: The travel time at quantile [Qi].

The Time, Duration, Average, Median and quantile values are formatted like HH:MM:SS (so hours, minutes and seconds always have two digits). For instance: 09:14:00 or 17:00:04. Elements on the diagonal of the matrix of locations (the same from and to address) are not calculated (0 travel time) and thus not returned.

4.2.2 Mapping to unknown arcs

The IE&IS group from the TU/e has provided us with the full set of time dependent travel time distributions between the 32 locations $S_{provided}$, which are requested by Quintiq. These travel time distributions are for 5 time periods, which means $S_{provided}$ consists of $32 \cdot 31 \cdot 5 = 4960$ travel time distributions. As will be discussed in Chapter 7, our test instances will have 319 locations, called S_{all}, which means we need $319 \cdot 318 \cdot 5 = 507,210$ travel time distributions. Requesting all travel time distributions from TomTom is computationally intensive and not feasible in the scope of our research. Therefore, we will use the known distributions for arcs

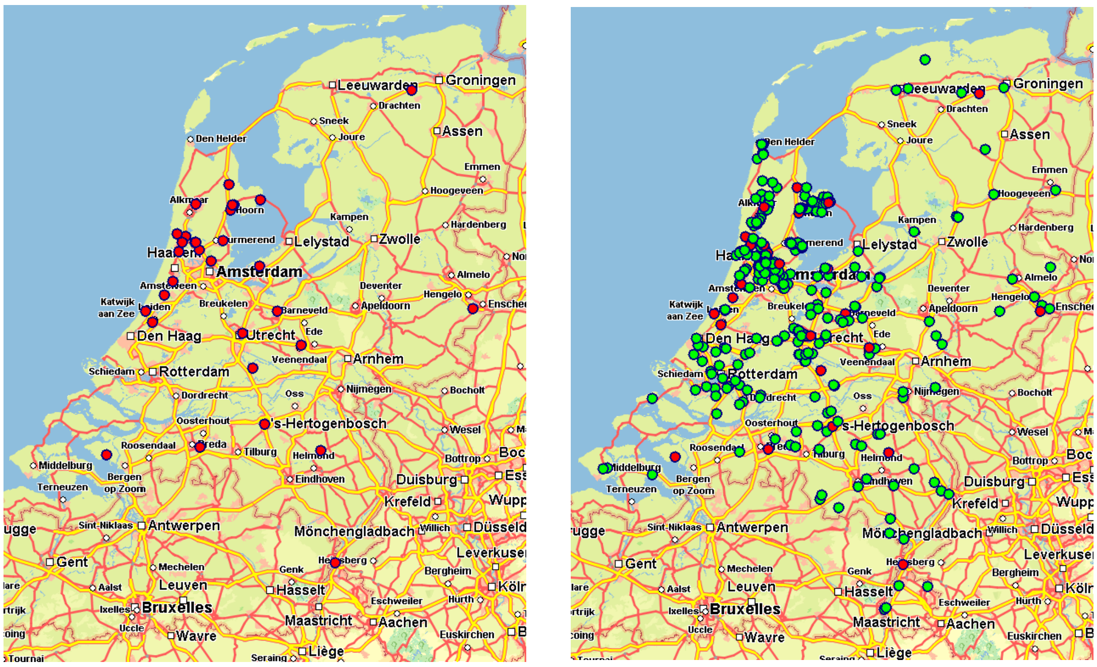


Figure 4.5: 319 locations: travel time distributions between all red locations in $S_{provided}$ are known. Travel time distributions between green locations in $\{S_{all} \setminus S_{provided}\}$ are not known.

for which we are missing a travel time distribution. We need two steps to get realistic travel time distributions for all arcs. The first step is assigning a travel time distribution shape to every arc. The travel time distribution shape expresses the uncertainty for an arc. The second step is to scale the travel time distribution such that the expected value matches the average travel time.

4.2.2.1 Getting a distribution shape for every arc

Figure 4.5 shows the locations in $S_{provided}$ in red and the locations in $S_{all} \setminus S_{provided}$ in green. We have the travel time distribution between every location in $S_{provided}$. We can map the travel time distributions for known arcs to arcs which are missing travel time distributions. When considering two arcs of which the from locations are close and the to locations are close to each other (as can be seen in Figure 4.2a), then they generally share a major part of the route. Looking to Figure 4.2a, if we have the travel time distribution for the arc from a to c, then we can use this distribution to approximate the travel time distribution for the arc from b to c. We use the shape of the distribution for the missing arc, but we should first scale the distribution such that the average of the new distributions equals the expected travel time of the arc.

If a location is not part of $S_{provided}$, then we map it using the Euclidean distance to the closest location from $S_{provided}$. After mapping every location in S_{all} to a location in $S_{provided}$ (every location in $S_{provided}$ maps to itself), we can use for every arc the travel time distribution between the mapped locations. We want to obtain the

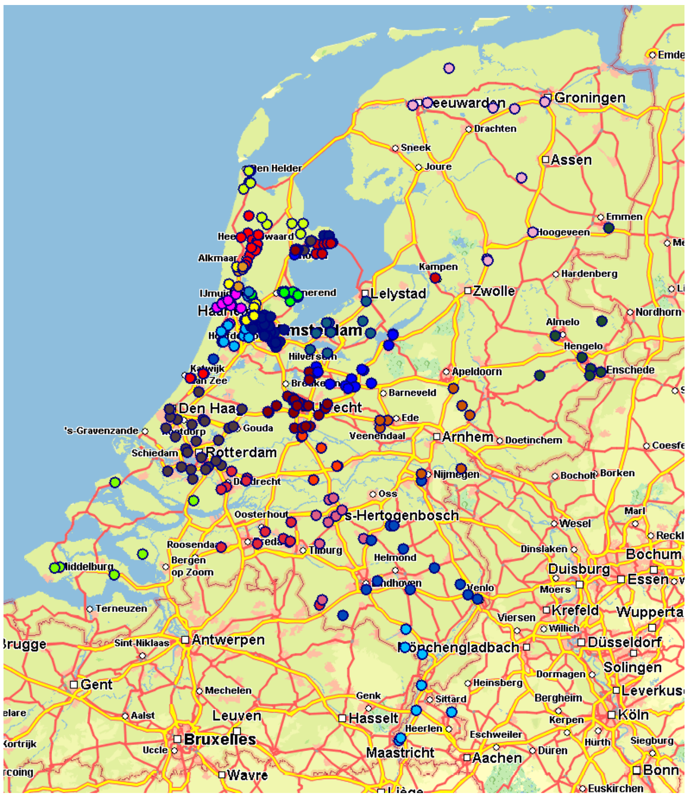


Figure 4.6: Locations with an equal color are mapped to the same location from $S_{provided}$

travel time distribution between locations a and b, which are mapped respectively to locations c and d. If a or b belong to $S_{provided}$, then $a = c$ or $b = d$.

c **is not equal to** d**:** both c and d belong to the set $S_{provided}$ because of the construction of c and d. This means we can use the travel time distribution shape from arc (c, d) for arc (a, b).

c **is equal to** d**:** because there is no distance between the locations, we do not have a useful travel time distribution for arc (a, b). Next we will explain how to handle this case.

Given location $a \in S_{provided}$, we define M_a as the set of locations which map to a. That is, for each location $c \in M_a$ the closest location $b \in S_{provided}$ to location c is location a:

$$\forall a, b \in S_{provided} : \forall c \in M_A : distance(c, a) \leq distance(c, b) \tag{4.6}$$

Figure 4.6 shows an overview where the locations in S_{all} are mapped to: locations with the same color are mapped to the same location. Set $S_{provided}$ has 32 locations, which means there are 32 groups of locations which have the same color. The largest set M_a has 53 locations (around Amsterdam), the second largest set has 26 locations (The Hague, Rotterdam), then 24 (Utrecht) and then 21 (Limburg and Eindhoven) locations.

4.2.2.1.1 Mapping in a cluster

Given locations a and b which are both not part of $S_{provided}$, that is, we do not yet have the travel time distribution between a and b. Assume they both map to location $c \in S_{provided}$. This means they are in the same geographical region, since otherwise they would have been mapped to different locations. For example, a and b can be in the same city when M_c is dense or they are in the same province when M_c is less sparse.

To get a travel time distribution between a and b we have request additional travel time distributions to the IE&IS group. To capture the characteristics of travel times in the region of a and b, we can request additional travel time distributions to characterize the travel times between locations of M_c. When we expect large differences in travel time distribution, we can increase the number of travel time distributions which are requested to get a better approximation. We have requested two travel time distributions for every location $a \in S_{provided}$. Given the set of addresses M_c which map to the same location c, there are $n \cdot (n-1)$ arcs between the locations. We sort these arcs according to expected travel time from PTV (system used in the DAIPEX application to obtain deterministic travel times between locations), after which we select a number of arcs which together characterize the travel times in this set well. Define A_{add} as the set of arcs which are additionally requested for set M_c. We have used two additional arcs at the 25th percentile and the 75th percentile. We have requested the travel time distribution for those two arcs from the IE&IS group.

We will use the additional distributions for the arcs between the locations in the set M_c. For an arc (a, b), where locations a and b are both part of M_c, we select the closest arc from A_{add} with relation to expected travel time. We then use the travel time distribution distribution from the selected arc for the arc (a, b). In the next section we will explain how to scale the distribution such that the expected travel time from the distribution equals the expected travel time of the arc.

4.2.2.2 Scaling travel time distributions

In the previous section we have seen how to obtain a travel time distribution for every arc given all travel time distributions for a subset of locations. We cannot use the travel time distributions directly, since the expected value of the distributions do not match the expected travel times for the arcs. Therefore, we will only use the shape of the distribution which expresses the amount of uncertainty on the travel time for the arc. The only information that is available about an arc for which we do not have the travel time distribution is the expected travel time which is used by the DAIPEX application. We scale the distribution such that the expected value of the distribution is equal to this expected travel time.

We use the Geographical Information System (GIS) provider PTV [1] in the DAIPEX application. The PTV system provides a distance matrix from which the distance between locations and the travel time between them can be retrieved. The travel times retrieved from PTV are based on trucks, because we are using trucks in the

[1] `http://xserver.ptvgroup.com/en-uk/products/ptv-xserver/ptv-xdima/`

Start period	End period	Expected value travel time distribution	0.05	0.10	...	0.95
00:00	07:15	0:44:10	0:33:53	0:35:31	...	0:58:42
07:15	09:30	0:59:53	0:36:06	0:37:53	...	1:55:02
09:30	17:00	0:45:47	0:35:08	0:36:36	...	1:01:01
17:00	19:00	0:54:03	0:35:11	0:36:46	...	1:40:52
19:00	23:59	0:42:43	0:33:37	0:35:15	...	*0:56:27*

Table 4.1: Travel time distributions from Zaandam to Utrecht

test instances as will be explained in Chapter 7. The first observation from the received travel time distributions is that the travel times are shorter than the travel times retrieved from PTV. The reason for this is that the travel time distributions received from the IE&IS group are based on both trucks and passenger cars and passenger cars drive faster than trucks in general.

In Chapter 5 we discuss solution approaches for the DTPP, but it is important to know that we use the travel times from PTV in varies solution approaches. To get a fair comparison between the solution approaches, the expected travel time from the travel time distributions should be equal to the travel times from PTV. The scaling is done in two steps. The first step is to scale a travel time distribution to get a relative distribution. This relative distribution expresses the dispersion for an arc and can be used for other arcs which suffer from equal traffic flow. The second step is to scale the relative distribution such that the expected value of the distribution equals the expected value of the arc.

We start with scaling to a relative distribution. Table 4.1 shows the travel time distribution received from the IE&IS group between Zaandam and Utrecht, including the corresponding expected value of the distribution which is provided as well. The distribution has a different shape for every time period. The different shapes become clear when scaling the distributions for the five time periods to a relative distribution.

Table 4.2 shows the relative travel time distributions. The travel time for every quantile is scaled using the expected value of the travel time distribution. For example, the travel time for the 95th quantile in the fifth time period (italic in Table 4.1) is 0:56:27 hours and the average of the time period is 0:42:43 hours. Dividing these travel times gives a dispersion measure about the distribution. A large value means the 95th quantile is far from the expected value, indicating there is much dispersion in the travel time. Converting the travel times to seconds gives a value of $\frac{0:56:27}{0:42:43} = \frac{3387}{2563} = 1.3215$.

After calculating the relative travel time distribution for every time period, we have to scale the distributions such that the expected value matches the expected travel time for this arc from PTV. We distinguish between time-dependent travel times

Start period	End period	Expected value travel time distribution	0.05	0.1	...	0.95
00:00	07:15	0:44:10	0.7672	0.8042	...	1.3291
07:15	09:30	0:59:53	0.6028	0.6326	...	1.9210
09:30	17:00	0:45:47	0.7674	0.7994	...	1.3327
17:00	19:00	0:54:03	0.6509	0.6802	...	1.8662
19:00	23:59	0:42:43	0.7870	0.8252	...	*1.3215*
Average		0:49:19				

Table 4.2: Relative travel time distributions between Zaandam and Utrecht

and a single time period for the day.

4.2.2.2.1 Time-dependent When using time-dependent travel times, the relative travel time distributions from Table 4.2 give the wrong mutual traffic flow. We have scaled all averages to their own expected value and we did not take into account that the expected values of the different periods are not equal. In other words, the relative distribution only tells something about the dispersion within one time period, while we are interested in the dispersion in relation to the 'normal' expected travel time from PTV.

To take the relation between the different time periods into account, we calculate a factor for every time period. This factor expresses how much the expected travel time for a time period is in relation to an overall average. For example, a factor of 1.2 for the second time period (the morning rush) has a meaning that a travel in the second time period takes on average 20% longer than the average travel time over the complete day. Note that this factor does not express anything about the dispersion of the distribution within a time period. The factor is calculated as the expected value of a time period divided by the average of the expected values of the five time periods.

The travel time distributions from Zaandam to Utrecht are provided (part of $S_{provided}$). We want to know the travel time distributions between Zaandam and Nieuwegein. Using the mapping as explained in Section 4.2.2.1 we find that the arc between Zaandam and Nieuwegein maps to the arc between Zaandam and Utrecht. Table 4.3 shows the expected values of the five time periods for two different arcs and the scaled expected values.

We calculate the time period factor for the fifth time period (evening hours), which is shown italic in Table 4.3. We divide the expected value of the fifth distribution by the average of the five distributions after converting the travel times to seconds: $\frac{0{:}42{:}43}{0{:}49{:}19} = \frac{2563}{2959} = 0.8661$. This means that a travel during the fifth time period takes on average 13.39% less than the average of the day.

Start period	End period	Expected value Zaandam to Utrecht	Time period factor	Expected value Zaandam to Nieuwegein
00:00	07:15	0:44:10	0.8955	1:05:14
07:15	09:30	0:59:53	1.2142	1:28:27
09:30	17:00	0:45:47	0.9283	1:07:38
17:00	19:00	0:54:03	1.0960	1:19:50
19:00	23:59	0:42:43	*0.8661*	1:03:06
Average		0:49:19	1.0	1:12:51

Table 4.3: Scaling the expected values of the time periods for the arc from Zaandam to Nieuwegein

Start period	End period	Calculated expected value travel time distribution	0.05	0.10	...	0.95
00:00	07:15	1:05:14	00:50:03	00:52:28	...	01:26:42
07:15	09:30	1:28:27	00:53:19	00:55:57	...	02:49:55
09:30	17:00	1:07:38	00:51:54	00:54:04	...	01:30:08
17:00	19:00	1:19:50	00:51:58	00:54:18	...	02:28:59
19:00	23:59	1:03:06	00:49:39	00:52:04	...	01:23:23

Table 4.4: Calculated travel time distributions from Zaandam to Nieuwegein for five time periods

The time period factors are shown in the fourth column of Table 4.3. To calculate the expected value of the five distributions for the arc from Zaandam to Nieuwegein we multiple the time period factor with the expected travel time provided by PTV. This provided us with expected values for the distributions of the five time periods. We can now scale the relative distributions to get a distribution which has the calculated expected values as expected value.

The last scaling step is multiplying the relative distribution with the calculated expected value of the distribution. Table 4.4 shows the final travel time distributions for the arc from Zaandam to Nieuwegein.

4.2.2.2.2 Single time period In the previous paragraph we have seen how to scale travel time distributions for time-dependent travel times. We also want to do experiments using non-time-dependent travel times, i.e. a single time period for a day. We have selected the morning rush which we will use for all day, because the travel time distributions for this time period have the most dispersion.

If we use the method for the time-dependent travel times also for a single time period, then the expected value of the distribution is for most arcs not equal to the expected travel time from PTV. To be more specific, for most arcs the expected

Start period	End period	Calculated expected value travel time distribution	0.05	0.10	...	0.95
00:00	23:59	01:12:51	00:43:55	00:46:05	...	02:19:57

Table 4.5: Calculated travel time distribution from Zaandam to Nieuwegein for one time period

value of the distribution will be higher than the expected travel time from PTV, since we have seen that the expected value of the morning rush period is in general higher than the average of the expected values. When using a single time period this is undesired: some solution approaches make use of the expected travel time from PTV and this influence the evaluation of the solution approaches.

Therefore, we use a different technique for scaling the travel time distribution when using a single time period: we ignore the time period factor in this case. This means we first scale to the relative travel time distribution shown in Table 4.2. We do not multiplying the expected travel time from PTV with the time period factor. Instead we multiply the expected travel time from PTV directly to the travel times of the different quantiles. The resulting distribution is shown in Table 4.5.

4.2.2.2.3 Discussion Using this mapping and scaling method, we have made several assumptions:

- Travel time distributions for trucks are similar to the provided travel time distributions which use a mixture of trucks and passenger cars.
- The relation between time periods is equal for the provided arc and the new arc.
- The dispersion in a time period is equal for the provided arc and the new arc. That is, the distribution shape is equal for the two arcs.

The combination of mapping distributions and scaling distributions has effect on the validity of the triangle inequality between the locations. When using independent sampling the triangle inequality is not satisfied. We improved the sampling method by using sampling based on hierarchical clustering. Nevertheless, the triangle inequality can still be violated. This can also happen in reality: a traffic jam incurs a long travel time, while taking a detour route can result in a shorter travel time. The resulting simulation worlds might not be perfect, but we believe that given the available information we constructed realistic simulation worlds. This can be improved by using a triangle fixing algorithm. In the available time of this research it proved to not be feasible to implement this, but Brickell, Dhillon, Sra, and Tropp (2008) solve the metric nearness problem using an iterative algorithm: given a set of distances between locations, find a 'nearest' set of distances that satisfy the triangle inequality. This allows to either make a significant change to a few distances or to change all distances a little.

5

Optimization algorithms to solve the DTPP

This chapter presents a number of optimization algorithms to solve the DTPP. Section 5.2 explains several ways on how to use slack to take stochastic travel times into account. We basically help an optimization algorithm using deterministic travel times by telling it to arrive some time before time windows close or by telling it that travel times are longer than they actually are expected to be. Section 5.3 explains how to use the Sample Average Approximation Approach. This technique uses simulation worlds by sampling the travel time distributions and taking those simulation worlds into account in the optimization algorithm.

We start by describing the optimization algorithm we use across this chapter in Section 5.1. That algorithm is based on the optimization algorithm of Logistics Planner, one of the Quintiq Products, and uses deterministic travel times. This algorithm is used in Section 5.2 and extended in Section 5.3.

5.1 Optimization algorithm

The first step of the optimization algorithm is clustering the orders which is explained in Section 5.1.1. After finishing the clustering a Large Neighborhood Search step is done, which is explained in Section 5.1.2.

5.1.1 Clustering

The first step in the algorithm is clustering of orders. The goal of this clustering is to get a good start solution for the Large Neighborhood Search, where the idea is that a cluster of orders 1) should fit together on a route and ii) should be close together such that one can create a route with limited distance. Directly after the clustering, the orders are sequenced on the route, which means the sequence of the orders on the route is determined taking into account the constraints and the Virtual Stochastic Costs as explained in Section 3.4.4.

The clustering algorithm uses multiple steps. The first step is estimating the number of routes needed based on an estimate of the number of orders per route. The second step, which is repeated multiple times, selects a subset of the routes and finds the best clusters for the orders which are planned on the selected routes.

The clustering algorithm is an iterative Mixed Integer Program (MIP). It uses an order anchor for a route which can be thought of as an order which represents the route: a driver drives to the stop location of the order anchor and the stop locations of the other orders on the route should be in the neighborhood of the stop locations of the order anchor. In every iteration the MIP takes a set of routes and minimizes the distance of the orders in the route to the order anchor.

Selecting the routes is done starting from a single route and then selecting routes in the neighborhood such that orders can be planned on another route where they might fit better. We randomly choose a route r which has orders planned on it. From route r we randomly select a number of orders O such that the number of selected orders equals the number of desired routes of the MIP iteration. For every order o_1 in O we select the order o_2 from the orders to cluster which is closest in terms of distance. We use the route on which order o_2 is planned in the current iteration and continue with selecting more routes until we have selected the desired number of routes.

The clustering in the MIP is done by a minimization goal which is twofold. The first subgoal is the distance between the depot and the stop location of the order anchor. The second subgoal is a metric measuring the compactness of a cluster. The cluster cost is defined as the combination of these two subgoals and the goal is to minimize this cost. The idea of the goal is to get routes with orders which fit well together on the route. There is also a penalty for violating the capacity constraint of the vehicle.

5.1.2 Large Neighborhood Search

After doing the clustering and sequencing the orders on the route, the Large Neighborhood Search (LNS) starts. The LNS method has two levels: at the top level there are iterations and in every iteration a subset of the problem is smartly selected for improvement, as explained in Section 5.1.2.1. This subproblem is improved in the second level using the Path Optimization Algorithm (POA), a Large Neighborhood Search framework implemented by Quintiq and explained in Section 5.1.2.2. A subproblem is independent of the rest of the problem, which makes it possible to improve multiple subproblems in parallel. We call the top level the LNS meta layer which has meta layer iterations and the second level is the POA layer and has POA iterations.

5.1.2.1 Subproblem selection

The subproblem selection starts by selecting an order, which we call the *leading order*, which will be used to select other orders which potentially fit well together on a route. The leading order is selected based on a hierarchical selection criterion. We first select orders which have not been in optimization the longest time or have not been selected as leading order the longest time. We then prefer orders which are not serviced in time and orders which are not yet planned.

After selecting a leading order, we select orders and routes to define the subproblem. We would like to have a subproblem such that POA converges to a local optimum. When using too many routes the search space becomes too large such that the search is too random or takes too long to converge. If the leading order is already planned, we select the route and all orders that are planned on this route. We randomly select orders which are nearby the order anchor. For a randomly selected order o, we also select the orders which are planned on the same route on which order o is planned. If there are unplanned orders or if there are routes which violate the capacity constraint we select additional routes which are preferably empty.

The LNS meta layer can select multiple subproblems in parallel for optimization to speed up the optimization algorithm. This is possible because the subproblems are independent from each other. As explained in this section, we select orders and routes which potentially fit well together. To get a good set of routes for every iteration, the number of routes in optimization is limited. As such a low number of routes will limit the number of concurrent POA invocations.

After selecting a subproblem, this subproblem is improved using the Path Optimization Algorithm as explained in the next section.

5.1.2.2 Path Optimization Algorithm

Quintiq has implemented a Large Neighborhood Search framework called the Path Optimization Algorithm (POA). When using POA one defines a set of nodes which are sequenced on paths taking into account the constraints and minimizes the score of the solution expressed by expressions. These are the different components of POA:

Nodes : The units that need to be planned. Every order will be a node in our problem.

Paths : Where the nodes are planned on. Every truck and trailer combination will be a path in our problem.

Expressions : These are needed to model constraints and goals. An expression is an entity such as distance or time. The value of the expression changes along the nodes of the path. The value change corresponds to a node or corresponds to the transition when going from one node to another node. For example, the expression 'distance' changes when driving from one location to another location. A path p can have multiple expressions, but all expressions operate on the same sequence of nodes on p.

Constraints : Defines what is allowed in a solution and what is not allowed. Constraints can be hard (no solution may violate them) or soft (a violation of the constraint incurs a penalty). Example constraints include time windows and the driving limit.

Goal : Expresses which solution is preferred if there are multiple solutions. The goal function can consists of a combination of weighted subgoals. An example goal is to minimize the total driving duration.

A problem defined by the components defined above is solved by POA using a Large Neighborhood Search (LNS) which is the second level of the LNS (the first level is the LNS meta layer). The LNS is defined by the POA strategy which has different actions. For example, a *destruction action* removes a number of nodes from the solution of the previous iteration and a *repair action* tries to insert unplanned nodes on the paths. Section 5.1.2.2.1 explains how we use the nodes, paths, etc. Section 5.1.2.2.2 explains the strategy to walk through the search space.

5.1.2.2.1 POA definition We explain the various components of POA as defined above.

Nodes and Paths A problem definition in POA starts with nodes which have to be planned on paths and sequenced on the path. Our problem includes only orders which either have to be loaded at the depot and delivered to a customer or picked up at a customer and unloaded at the depot. The sequence of the orders at the depot is not relevant, since there is no driving between the loading or unloading of orders at the depot. Therefore, we use a single node for every order which represents the load or unload action at the stop location outside the depot.

An order is planned on a route, which is a combination of resources: a truck, a trailer and a driver. A route is represented in POA by a path, so we create a single path for every route.

Expressions The next POA components are the expressions. We have the following expressions:

Capacity This expression represents the loaded amount in the vehicle. When loading an order the value increases and when unloading an order the value decreases with the amount that is (un)loaded. The value remains equal during driving and waiting.

Time This expression represents the time. A change in the time expression corresponds to loading and unloading orders, to driving between two locations or to waiting time.

Distance This expression represents the distance. The value changes when driving from one location to another location.

Constraints We have multiple constraints to define the problem in POA.

If a constraint is soft, then a violation incurs a penalty cost which is expressed by the following formula:

$$Penalty\ costs = factor \cdot (offset + violation)^{power} \tag{5.1}$$

The $factor, offset$ and $power$ are parameters of the algorithm. They should be set such that if there are two different solutions S_1 and S_2 and the total score for S_1 is lower than the total score for S_2, then solution S_2 should indeed be worse than solution S_1.

The first constraint is the capacity constraint, which expresses that every resource has a maximum capacity which can be loaded at the same time. This is a soft constraint meaning that violating this constraint incurs a penalty according to Formula 5.1.

The second constraint enforces time windows. An order cannot be serviced before the release date. This is a hard constraint, that is, no solution can have the service of the order start before the release date. If a driver arrives at a location before the release date, the driver has to wait until the release date. The due date of a time window is a soft constraint meaning that violating this constraint incurs a penalty according to Formula 5.1.

The third and last type of constraint is related to driving regulations. As explained in Section 3.3.5, a driver is allowed to drive 4:30 hours after which a break of 45 minutes is needed. POA internally uses calendars to enforce that after 4:30 hours a break of 45 minutes is planned in which no other actions can occur. We allow to plan a break of 45 minutes if the waiting time caused by a time window is at least 45 minutes.

Goal The last component is the goal function, which is build up from various components. The distance expression is used to express that every driving kilometer costs money. The time expression is used to express the costs for driving, waiting, loading, and unloading. We use a single cost per hour for these four activities which will be explained in Section 7.1. The distance and duration together give a total goal score. We combine this with the total constraint score for the different constraint to get the a Total Score. This score is used by POA to evaluate solutions.

5.1.2.2.2 POA strategy After the POA problem is constructed, the strategy defines how the search space is explored with the available search actions. They can be categorized in *destruction actions* and *repair actions.*

We use three destruction actions: stop destruction, area destruction, and path destruction. They have an execution chance such that every iteration different nodes are removed from the existing solution. Nodes that are next to each other on a path and that have the same location are considered to belong to the same stop. When stop destruction is executed, all nodes from one randomly selected stop are unplanned. When area destruction is executed, a stop is selected at random after which the nodes corresponding to multiple nearest stops are unplanned. When path destruction is executed, a path is randomly selected and all nodes at this path are unplanned.

Besides destruction actions, we also have a repair action to recreate solutions using the previously unplanned nodes. The repair action randomly selects an unplanned node for which all possible *moves*, a possible way to plan the node, are determined. Each move has an estimate of how good it is, and we place the best moves in a *population*. Only the moves in the population are completely calculated such that the move with the best score is selected. The best move is executed by planning the corresponding node in the sequence of the corresponding path. The size of the population is adaptive: if a move with a relatively bad estimate is often chosen as best move, then the population size grows. The population size shrinks if the move with the worst estimate is rarely chosen.

The POA strategy uses *parachutes* to get initial solutions. These are highly random solutions, obtained by unplanning a large part of the orders in the initial solution and then randomly planning them.

POA starts with all solutions in the solution pool generated by the parachute landings and improves the solutions. During the execution of POA we maintain the *search depth*. For every search depth the solutions in the solution pool are improved. Improving a solution is done by executing the defined search actions. The search actions which are defined before are executed until either a certain amount of iterations has been done or until a certain amount of time has passed. After the search actions are executed the best solution is added to the solution pool and this is used in the next search depth.

When the strategy has finished, the best solution is returned to the LNS meta layer The meta layer evaluates the provided solution, writes the solution to the model and starts a new meta layer iteration if the optimization duration has not been met.

5.2 Building in slack

By building in slack in a transportation plan we want to try to guarantee more orders are serviced within their service reliability requirement. Slack can be built in the transportation plan using different methods where we consider two types of slack: *time window slack* and *travel time slack*. When using time window slack we reserve time at the end of the time window, which we explain in Section 5.2.1. The idea here is that you plan to be at a customer some time before the time window closes to account for unforeseen longer travel times. The idea of travel time slack is to overestimate travel times, which we explain in Section 5.2.2. By means of this we make the optimizer always consider a worse scenario than the expected scenario. This means that if we solve the problem in a worse scenario, we likely also have a better solution when evaluating using stochastic travel times.

Kok et al. (2010) considered travel time slack when solving the vehicle departure time problem where the departure time from the depot has to be decided to minimize the duty time of the driver while satisfying time windows. They consider breaks in their problem and they propose to use travel time slack instead of explicit driver breaks scheduling. They show that adding slack to the travel times reduces the number of infeasible routes from 64% to 2%, but only for light congestion (with medium and high congestion the percentage of infeasible routes remains rather large). To the best of our knowledge, no further research has been reported on using slack to generate transportation plans when dealing with stochastic travel times.

Although a generated transportation plan is based on slack, we do not include slack in the final transportation plan which is communicated to the drivers. As discussed in Section 3.2, a transportation plan defines among others the time $std_{v,l}$ when a vehicle v starts driving from location l. In general the driver does not wait at locations, because waiting uses precious time which is not available to serve other customers. Therefore, the generated plan will not include explicit waiting actions, instead the actions are planned as soon as possible. Waiting which is the result of time windows not being open can still occur. It is also possible there are late orders, because we have not included enough slack.

To evaluate the quality of the solutions generated by the different optimization algorithms we use Sample Average Approximation as explained in Section 5.3.

5.2.1 Time window slack

The idea of time window slack is to reserve time at the end of each time window. By requiring to be at the location of an order a certain amount of time before the time window closes, we want to guarantee that the order will also be served on time when unforeseen changes in the travel times occur.

We require the optimization algorithm as explained in Section 5.1 to provide a transportation plan using the reduced time windows. This means the optimization algorithm generates a plan using fixed travel times and is not aware of stochastic travel times. Given an order o_1 with a time window of $[r_{o_1}, d_{o_1}]$ where we are using tws minutes time window slack, we require from the optimization algorithm to start the service of o_1 between r_{o_1} and $d_{o_1} - tws$. This means there will be a penalty in the optimization algorithm for a solution in which the start of the service of o_1 is in $[d_{o_1} - tws, d_{o_1}]$, which is on time according to the original time window.

Next to changing the time windows, we also take the driving limit into account when using time window slack. Instead of using the normal driving limit of 9 hours, we decrease the driving limit with the amount of time window slack.

$$drivingLimit_{adapted} = drivingLimit_{actual} - slack \tag{5.2}$$

After the optimization algorithm has finished, we evaluate the transportation plan using Sample Average Approximation (SAA) as explained in Section 5.3 where we do not use the slack and we use the original time window $[r_{o_1}, d_{o_1}]$. We hope to see that using the provided plan more orders are served within the service reliability requirement.

5.2.1.1 Small time windows

We can easily use the above approach of time window slack if the amount of slack is less than the size of the time window. If more slack than the time window size should be added, then the new due date is before the release date of the time window. This would always result in constraint costs, since the release date of the time window is a hard constraint. Therefore, we define the release date of the time window to be equal to the due date in this case. For example, for order o_2 with a time window of [9:59 - 10:00] and 30 minutes time window slack, we use the new time window [9:30 - 9:30] in the optimization algorithm.

Besides changing the time window, we also have to change the duration of the service to reflect both the introduced slack and the duration of the action. To prevent that the service of the order has been finished before the time window has actually started, we increase the duration of the service. We add the time it takes from the new release date until the original release date to the service duration.

We summarize the reduced time window, where *actual* refers to the time window before introducing slack and *adapted* refers to the adapted time window. The duration of the service is denoted by $duration_{actual}$.

$$d_{adapted} = r_{actual} - slack \tag{5.3}$$

$$r_{adapted} = min(r_{actual}, d_{actual} - slack) \tag{5.4}$$

$$duration_{adapted} = duration_{actual} + max(slack - size\ time\ window, 0) \tag{5.5}$$

If the service starts at the start of the reduced time window, the adapted duration ensures this corresponds to the action starting at the release date of the actual time window. The disadvantage of this method is that if the time window is smaller than the slack and the order is planned in its actual time window, then the service also includes the slack duration. The result is that too much time is planned in for the service of the order compared to the actual service duration. This may lead to better on time performance, but also to underutilization of vehicles.

For example, we use 30 minutes time window slack for order o_2 with a service duration of 10 minutes and with time window [9:59 - 10:00]. This gives an adapted time window of [9:30 - 9:30] and an adapted service duration of $10 + 30 - 1 = 39$ minutes. If o_2 is planned to start at 10:00 we still use the adapted service duration of 39 minutes giving an end time of 10:39, while the actual service ends at 10:10. The reason we are always using the adapted service duration is that the POA framework requires a fixed service duration and this cannot depend on the start time of the service.

5.2.2 Travel time slack

In the previous section we have seen how to use time window slack in the optimization algorithm. In this section we discuss travel time slack to overestimate travel times. The idea of using travel time slack is that when making a transportation plan we overestimate the travel times such that the orders have a higher reliability of being serviced on time. We consider travel time slack using a fixed percentage of the travel time (Section 5.2.2.1), travel time slack based on the standard deviation of the travel time distribution (Section 5.2.2.2), and travel time slack based on the Mean Absolute Deviation from the mean (MAD) of the travel time distribution (Section 5.2.2.3).

5.2.2.1 Travel time slack using a fixed percentage

When using the fixed percentage approach, we overestimate the travel time using the expected travel time. Given a *percentage*, with $percentage \geq 0$, we calculate the new travel time using the fixed percentage method with the following formula:

$$TravelTime_{fixedPercentage} = ExpectedTravelTime \cdot (\frac{percentage}{100} + 1) \qquad (5.6)$$

This provides for every arc a travel time. We use the overestimated travel times in the optimization algorithm as described in Section 5.1. After the optimization algorithm has finished, we evaluate the transportation plan using Sample Average Approximation (SAA) where we do not use travel time slack.

5.2.2.2 Travel time slack based on Standard Deviation

In Sections 5.2.1 and 5.2.2.1 we have seen two approaches how to use slack which is independent of the stochastic travel times. The approaches do not take into account that for some arcs there is more variability in the travel times than for other arcs. In this and the next approach the slack does depend on the stochastic travel times.

When using travel time slack based on the standard deviation, we use the standard deviation of the travel time distribution to calculate the slack for every arc. We use the Sample Average Approximation method as explained in Section 5.3 to get a set of N simulation worlds. Recall that $Q^p_{i,j}(q)$ is the quantile function of the travel time defined in Section 4.2 which expresses that given a quantile q the travel time from location i to location j in time period p is $Q^p_{i,j}(q)$. In Section 4.1.2 we introduced the sampling method where $q^s_{i,j}$ represents the sample for simulation world $s \in S$. We calculate for every arc (i, j) the slack using the simulation worlds. We have to select a time period to calculate the standard deviation. If we have probability distribution which covers the full day, we can use this to calculate the slack. If the travel times are time-dependent, we select the time period which has the highest variability in travel times to calculate the standard deviation.

The expected travel time for arc (i, j) is denoted by $ExpectedTravelTime(i, j)$ and the overestimated travel time for arc (i, j) by $TravelTime_{std}(i, j)$. The standard deviation tells a lot about the variability of the travel time, but adding this as slack might be too much or not enough. Therefore, we introduce a factor f_{std} which we use to scale the standard deviation. This factor is arc independent (i.e. we use one factor for all arcs) and several alternatives are tested in Chapter 7. We use the sample standard deviation, because the travel times are a sample of the full population of travel times.

$$\overline{Q}^p_{i,j} = \frac{1}{|S|}\sum_{s\in S} Q^p_{i,j}(q^s_{i,j}) \quad \textit{// average travel time on arc(i,j)} \tag{5.7}$$

$$\sigma_{i,j} = \sqrt{\frac{1}{|S|-1}\sum_{s\in S}\left(Q^p_{i,j}(q^s_{i,j}) - \overline{Q}^p_{i,j}\right)^2} \tag{5.8}$$

$$TravelTime_{Std}(i,j) = ExpectedTravelTime(i,j) + \sigma_{i,j}\cdot f_{std} \tag{5.9}$$

After the optimization algorithm has finished, we evaluate the transportation plan using Sample Average Approximation (SAA) where we do not use travel time slack.

5.2.2.3 Slack based on MAD

In the previous section we have seen how to use the standard deviation to overestimate travel times. Instead of using the standard deviation we also consider the Mean Absolute Deviation from the mean (MAD) as proposed by Postek, Ben-Tal, Den Hertog, and Melenberg (2015) in their research on Robust Optimization.

The calculation of the MAD is similar to the calculation of the standard deviation, but instead of squaring the differences we take the absolute values for the differences and we do not take the root:

$$\overline{Q}^p_{i,j} = \frac{1}{|S|}\sum_{s\in S} Q^p_{i,j}(q^s_{i,j}) \quad \textit{// average travel time on arc} \tag{5.10}$$

$$MAD_{i,j} = \frac{1}{|S|}\sum_{s\in S}\left|Q^p_{i,j}(q^s_{i,j}) - \overline{Q}^p_{i,j}\right| \tag{5.11}$$

$$TravelTime_{MAD}(i,j) = ExpectedTravelTime(i,j) + MAD_{i,j}\cdot f_{MAD} \tag{5.12}$$

After the optimization algorithm has finished, we evaluate the transportation plan using Sample Average Approximation (SAA) where we do not use travel time slack.

5.2.3 Travel time slack in combination with breaks

The idea of using travel time slack is that when making a transportation plan we overestimate the travel times such that the orders have a higher reliability of being serviced on time. What we expect when selecting a route and comparing the route with and without slack, is that the route without slack will always be earlier or at the same time at every customer location. Every arc in the overestimated route has at least the same travel time as without using slack. However, when including breaks it is no longer evident that the route with slack is always at least as late at any location than the route without slack which we will discuss in this section.

In general a route is planned such that the driver arrives exactly on time at many customer locations: waiting incurs costs so this should be prevented if possible. This also applies to the route which is planned using travel time slack. The vehicle v_{slack} in the route with slack arrives mostly in the time window of the order such that no waiting is needed before starting the service. When using the route without slack, the travel times are shorter and the vehicle $v_{noSlack}$ can arrive before the release date of the order, in particular when the time windows are tight. This causes waiting time for vehicle $v_{noSlack}$, while vehicle v_{slack} was driving during that waiting time. This waiting time is often less than 45 minutes, so no break can automatically be planned in the waiting time, as this would delay the start of the next service.

Because we plan a break after 4:30 hours driving, vehicle v_{slack} has its break earlier on the day than vehicle $v_{noSlack}$. The result is that vehicle $v_{noSlack}$ needs a break later on the day after 4:30 hours driving. This can cause a delivery to be late for vehicle $v_{noSlack}$ while it was on time for vehicle v_{slack}. This is not what one would expect and can be solved by using the breaks from vehicle v_{slack} also for $v_{noSlack}$. This means that the vehicle takes a break earlier than required, but it prevents the situation that a delivery is late while it was not when considering slack.

5.3 Sample Average Approximation

In Section 5.2 we have shown how to include slack in different ways into the optimization algorithm as described in Section 5.1. The resulting optimization algorithm is working with fixed travel times and does not explicitly consider the stochastic travel times. Verweij et al. (2003) discuss the Sample Average Approximation (SAA) method which is an approach for solving stochastic optimization problems by using Monte Carlo simulation. Using this approach we generate a set of simulation worlds which together approximate the travel time distributions. In Section 4.1.2 we showed how we sample in a smart way from the travel time distributions to generate simulation worlds. In this section we explain how to extend the optimization algorithm described in Section 5.1 to solve the DTPP by explicitly taking stochastic travel times into account. We first explain how to incorporate Sample Average Approximation in the DAIPEX application (Section 5.3.1), after which we explain how to include it in the optimization algorithm (Section 5.3.2).

5.3.1 Sample Average Approximation in the DAIPEX application

To evaluate the quality of a transportation plan we use the Sample Average Approximation (SAA) approach. We also refer to an optimization algorithm using SAA as a travel time distribution optimizer, since it takes the distributions into account in the optimization. For every order in every simulation world we have to evaluate if the service starts on time. We maintain for every order on every route for every simulation world a vector expressing the *time logic*. The time logic of an order gives all details about the order related to time. The vector stores among others the start time of the service, the break duration since the last break, the driving time since last break, and the total driving time on the day. The vector is initialized for the first order on the route. For every next order we can calculate the time logic vector based on the vector of the previous order. The time logic calculation includes enforcing the release dates of the time windows, planning breaks (using Algorithm 2), and maintaining the total driving time.

The time logic vectors can be used to calculate the service reliability for every order. The time logic as calculated in the DAIPEX application should be equal to the time logic in the optimization algorithm, which is explained next.

5.3.2 Sample Average Approximation in optimization algorithm

We have seen the POA definition in Section 5.1.2.2.1. This includes a single expression to represent the time. In the solution approaches where we use fixed travel times a single expression is enough to represent the time entity. However, when using Sample Average Approximation (SAA) we have to maintain the time for every simulation world. We introduce an expression $Time\_s$ for every simulation world $s \in S$ with the corresponding travel times. In Section 5.3.2.1 we explain how to get the correct start times from the depot for every simulation world. We explain in Section 5.3.2.2 how to evaluate the service reliability requirement for the DTPP and in Section 5.3.2.3 we explain how to implement the driving limit reliability requirement. We end in Section 5.3.2.4 with an explanation how to implement breaks for time-dependent stochastic travel times.

5.3.2.1 Enforce start time from depot

In the POA framework nodes can be planned either *as-soon-as-possible* (ASAP) or *just-in-time* (JIT). Planning nodes ASAP in combination with a variable start time of the route from the depot has as result that there is much unwanted waiting which incurs costs. Planning nodes JIT causes a different problem as shown in Figure 5.1. The vehicle departs from the depot at 6:30 when using the expected travel time to exactly arrive at the release date of the first order C_1. In the first simulation world the route starts later than 6:30, because the travel time is less than the expected travel time from the depot to the first location. In the second simulation world on the other hand, the route starts before 6:30, because the travel time is larger than the expected travel time. In these cases the driver had preliminary information on the stochastic travel times, while this is not possible in practice.

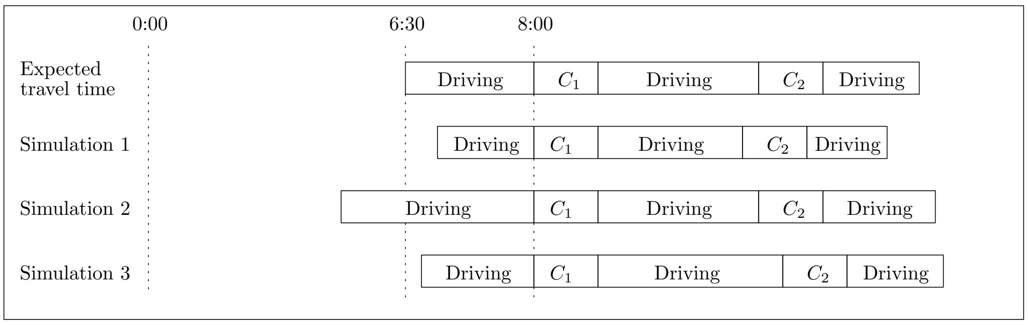


Figure 5.1: Planning nodes just-in-time in POA.

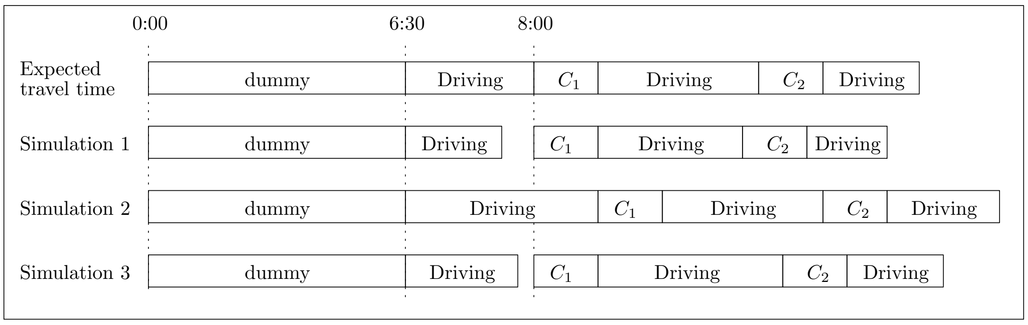


Figure 5.2: Using dummy nodes in POA in combination with ASAP planning to enforce correct start times from the depot in the simulation worlds.

We use *dummy nodes* in POA to enforce correct start times from the depot in the simulation worlds. Figure 5.2 shows that these dummy nodes 'block' the time until the departure time from the depot using the expected travel time. The driving nodes are planned ASAP, such that in the first simulation there is waiting time due to shorter driving time and in the second simulation world the service of order C_1 starts later.

5.3.2.2 Service reliability requirement

When using fixed travel times we enforce time windows of orders by using a hard constraint for the release date of the time window and a soft constraint for the due date with penalty costs according to Formula 5.1. If we are using SAA we can use the simulation worlds to evaluate the service reliability of orders using the Service Costs as defined in Section 3.4.3. The POA framework has a constraint (*MultiEndConstraint*) which allows to express that the service reliability of each order should meet at least a required lower bound: the service of an order should start before the due date of the order in at least $lowerbound \cdot |S|$ simulation worlds with $0 \leq lowerbound \leq 1$, and S the set of simulation worlds. For order o the violation and reliability are defined as follows. If the violation is positive, it can be used in Formula 5.1 to calculate the penalty costs for the service reliability requirement. If the violation is zero, the penalty costs are zero.

$$reliability_o = \frac{number\ of\ expressions\ on\ time}{total\ number\ of\ expressions} \tag{5.13}$$

$$violation_o = max(lowerbound - reliability, 0) \tag{5.14}$$

The release date of a time window is expressed using a hard constraint for every time expression. When using Sample Average Approximation we still have the time expression with the expected travel times, because we use the expected travel time for the departure time from the depot. This can be improved as explained in Chapter 6.

The POA framework has the *EndConstraint* and the *InTimeConstraint*. The former expresses that if there are multiple orders for the same location (a stop in POA), then every order must be served on time, while the latter expresses that only the first order of the stop should be served in time if the other orders follow directly. The Logistics Planner optimization algorithm normally uses the InTimeConstraint to evaluate lateness of orders. The POA constraint which we use to express the service reliability requirement is the *MultiEndConstraint*, because there is no *MultiInTimeConstraint* available in the POA framework. Therefore, we have also changed this in the DAIPEX application to use the EndConstraint instead of the default InTimeConstraint. Note that we advice Quintiq to implement the MultiInTimeConstraint.

5.3.2.3 Driving limit

The optimization algorithm of Logistics Planner includes functionality for driving regulations. We have already seen in Section 5.1.2.2.1 that we use the available driving breaks. Next to driving breaks the DTPP has a driving limit which can be modeled using night rests. But activating night rests while in fact drivers have a maximum driving duration of 9 hours has a negative effect on the performance of POA. For example, when the driving limit is violated with one minute, a night rest of 11 hours is added. There is only little difference in costs between violating the driving limit with 1 minute or with one hour, because the night rest of 11 hours is dominant in the costs. It is better to have a gradual cost function where the amount of violation is taken into account for the constraint costs. Therefore, we disabled the night rests.

An alternative approach we have considered is using the *EndPathConstraint*, which is normally used for maximum route duration. However, there are several problems with this approach. First of all, breaks are included in the route duration, while the driving regulations state that you can only drive 9 hours a day excluding breaks. Secondly, the dummy nodes introduced in Section 5.3.2.1 give an increased route duration while the dummy nodes do not belong to the driving time. Thirdly, the driving limit reliability requirement requires an EndPathConstraint which can be evaluated on multiple expression and this is not available in the POA framework.

To overcome the described problems, we have added an extra POA expression for the driving time. This expression suffices when using fixed travel times, because we can use a POA constraint to enforce the driving limit (*EndPathConstraint*). If we are using Sample Average Approximation, then the driving limit reliability requirement should be expressed on the driving times of all simulation worlds. Therefore, we need for every simulation world $s \in S$ next to the introduced time expression $Time_s$, an expression $DrivingTime_s$ for the driving time in simulation world s.

If we are using a single time period for the day, then the driving time expressions are correct when they only contain the driving time. We can use the MultiEnd-Constraint to express the driving limit reliability requirement. When we are using time-dependent travel times, it is not enough to only have the driving time on the expression, because the driving time depends on the time of breaks as we have seen before. This means we have to add the full time logic including service durations and time window constraints also on the driving time expressions. Directly applying the MultiEndConstraint is not possible due to the presence of service durations and waiting times. We solve this by enabling the night rests on the driving time expressions: using a calendar we can add a night rest of 2 days after reaching the driving limit of 9 hours. Instead of comparing the amount of violation, we use the MultiEndConstraint on the driving time expressions with an latest end of two days such that all expressions which violate the driving limit and thus take longer than two days are counted as a violated expression.

The large increase in number of expressions does have an influence on the performance of POA, which we will show in Chapter 7.

5.3.2.4 Breaks for Sample Average Approximation

In this technical section we explain how to implement breaks in POA in the simulation worlds when using time-dependent travel times. For breaks using a single time period we refer to Section 5.1.2.2.1 which should be applied for every simulation world. Planning breaks for time-dependent travel times is more complicated than breaks for a single time period as explained in Section 4.1.1. We have seen that the exact start time of the break is required, otherwise the travel time can be wrong.

In POA one can set a flag $WaitingResetsBreakCalendars$ which indicates that a driver is allowed to rest during waiting in case the waiting time is at least the break period. For this flag to work the expression path of each route should have the participation calendar be set.

Time-dependent travel times are set in POA using a *transition calendar*. One can specify for each period of the day which travel times should be used. This transition calendar is added to a *combined calendar* (adding the transition calendar to the Dynamic Transition of an expression does not work because then the break calendar is not aware of the transitions such that no breaks are planned) and the combined calendar is set as the active transition calendar of the expression path of a route. Setting this combined calendar also as the active participation calendar for this expression path, which is required to plan breaks during waiting as explained in the previous paragraph, is technically not possible because the combined calendar is not defined on participation level.

This can be solved by defining two combined calendars: the first combined calendar uses the transition calendar with the time-dependent travel times and the break calendar for the driving breaks and is set as the active transition calendar of the expression path of a route. The second combined calendar only uses the break calendar for the driving breaks and is set as the active participation calendar. It is important that one break calendar for the driving breaks is created and used in both combined calendars, because otherwise the two break calendars will both schedule breaks.

5.3.3 Virtual Stochastic Costs versus Virtual Expected Costs

In Section 3.4.4 we have introduced the Virtual Stochastic Costs. This objective expresses the costs of the vehicle and the driver, but also the penalty costs incurred by not meeting the service reliability requirement or driving limit reliability requirement. We use this objective to evaluate a transportation plan which is generated when using fixed travel times or when using the Sample Average Approximation in the optimization algorithm. In the latter case we can directly use the Virtual Stochastic Costs in the optimization algorithm to evaluate the transportation plan at intermediate steps.

We cannot use the Virtual Stochastic Costs when using fixed travel times at intermediate steps, because we do not use the Sample Average Approximation approach in this case. Therefore, we introduce the *Virtual Expected Costs.* This objective uses the costs for the vehicle and the driver as in the Virtual Stochastic Costs, but the penalty for constraint violations are calculated differently.

Instead of using the reliability requirements for the service and the driving limit, we use constraints for fixed travel times.

5.3.3.1 Service costs fixed travel times

We check whether the start of the service according to the expected travel time is before the due date of the order. Let $\mathbb{E}(\boldsymbol{S}_o)$ be the start of the service of order o according to the expected travel time. We define the violation for order o as $violation_o = \max(d_o - \mathbb{E}(\boldsymbol{S}_o), 0)$.

$$ServiceCostsFixed_o = \begin{cases} 0 & \text{if } violation_o = 0 \\ f_{late} \cdot (violation_o + vo_{late})^{p_{late}} & \text{if } violation_o > 0 \end{cases} \tag{5.15}$$

where f_{late}, vo_{late} and p_{late} are equal factors for all orders.

We use the $ServiceCostsFixed_o$ for order o to define the total service for a route using fixed travel times:

$$TotalServiceCostsFixed_{R_v} = \sum_{i=1}^{|R_v|} ServiceCostsFixed_{R_v} \tag{5.16}$$

5.3.3.2 Driving limit costs fixed travel times

Define $driving_R$ as the driving time in route R. Let $DrivingLimit$ be the driving limit according to European Regulations and define $viol_R = \max(driving_R - DrivingLimit, 0)$ as the amount of driving time above the driving limit. The driving limit costs are positive when there is a violation of the driving limit:

$$DrivingLimitCostsFixed_{R_v} = \begin{cases} 0 & \text{if } viol_{R_v} = 0 \\ f_{dl} \cdot (viol_{R_v} + vo_{dl})^{p_{dl}} & \text{if } viol_{R_v} > 0 \end{cases} \tag{5.17}$$

where f_{dl}, vo_{dl} and p_{dl} are equal factors for all routes.

5.3.3.3 Virtual Expected Costs

We define the *Virtual Expected Costs* as the sum of the costs for the vehicle plus the constraint costs for the service and driving limit using fixed expected travel times.

$$\begin{aligned} Virtual\ Expected\ Costs = \sum_{v \in V} (&CostsVehicle_{R_v} + CostsDriver_{R_v} + \\ &TotalServiceCostsFixed_{R_v} + \\ &DrivingLimitCostsFixed_{R_v}) \end{aligned} \tag{5.18}$$

We use this objective in the optimization algorithms when working with fixed travel times. When the algorithm is finished, we evaluate the generated transportation plan using the *Virtual Stochastic Costs*.

6

Start time calculation

In Section 3.2.1 we discussed the planning decisions which are needed to solve the DAIPEX Transportation Planning Problem. We have seen that our optimizer decides which orders should be transported by which vehicles and in which order the orders should be serviced. Next to these two planning decisions, we also have to decide on the start time of the route from the depot. Using the start time of the route we have influence on the service reliabilities of the orders. Hypothesis 7 states that using travel time distributions to calculate the departure time from the depot can increase the service reliabilities of orders. If the first order has a service reliability of 90% for a given departure time from the depot, while it requires 95%, we can leave earlier from the depot to increase the service reliability. This does not only apply to the first order of a route, but it can also apply to other orders in the route. If the orders have time windows with a release date during the execution of the route, then the departure time from the depot can have less influence on the service reliabilities. The release date of these time windows results in waiting time if a vehicles arrives before the release date. In this case departing earlier from the depot cannot increase the service reliabilities much.

Lekkerkerker (2016) proposed three algorithms to optimize the departure time from the depot. Their objective is twofold: the first objective is to service all customers with a service reliability which is as close as possible to the service requirements of the orders. Often there is a myriad of solutions which all have the same service reliabilities for the orders. Therefore, Lekkerkerker defined a second objective to get the lowest possible costs by leaving the depot as late as possible without decreasing a service reliability of any order.

The first objective of Lekkerkerker is a convex function of the departure time of a route: leaving the depot earlier can never decrease the service reliability of any of the orders due to the FIFO property (explained in Section 4.1.1). Given a latest start time of a route, the second objective is to minimize the costs. The second objective is also a convex function of the departure time of the route: leaving earlier than the given latest start time always increases the costs. The first algorithm proposed by Lekkerkerker is a binary search on the optimum departure time from the depot. This algorithm assumes that the objective function is convex. We will see that this

Departure time depot	5:00	6:40	6:50	7:04	7:10
Service reliability	76%	76%	75%	75%	73%
Virtual Stochastic Costs	1,789	1,716	1,720	1,709	1,731

Table 6.1: Virtual Stochastic Costs as function of the departure time

is not the case for the objective of the DTPP.

In Section 3.4 we defined the objective Virtual Stochastic Costs based on two parts: the costs of the vehicle and driver, and the costs for violating a constraint. There is a clear difference between the Virtual Stochastic Costs and the objective used by Lekkerkerker. Lekkerkerker first requires the highest possible service reliability at the expense of any costs of resources and only secondly the costs are minimized without lowering the service reliability. Our objective Virtual Stochastic Costs is a combination of the costs of resources and the constraint costs for not meeting the service reliability requirements. The motivation is that meeting all service reliability requirements should not be done at the expense of every costs.

Given a route, the Virtual Stochastic Costs is not convex in the departure time from the depot. For a fixed service reliability the driver can depart earlier from the depot, but departing earlier costs money due to an increase in waiting time (if there are time windows) and does not gain anything on the service reliability. The driver can depart even earlier in order to increase the service reliability of an order. The service requirement violation can decrease more than the increase of the costs due to more waiting time such that the Virtual Stochastic Costs decreases. Then the same argument holds that departing earlier incurs waiting time and thus an increase in costs while not gaining anything on the service reliability.

Table 6.1 shows an example route with different departure times from the depot. This table shows the Virtual Stochastic Costs of a route and the service reliability of the 8th order on this route. We assume all other orders on this route satisfy their service requirement. We see that leaving the depot at 5:00 gives a solution with a service reliability of 76% and a Virtual Stochastic Costs of 1,789. Departing at 6:40 gives the same service reliability of 76%, but the Virtual Stochastic Costs decreases due to a shorter working time (there was waiting when departing at 5:00). Departing even later at 6:50 gives 1% less service reliability and therefore the costs increase to 1,720. The same service reliability of 75% can be obtained by departing at 7:04 with a lower costs. Departing later causes a decrease of the service reliability of 2% to 73% and an increase in costs again. This example confirm that the Virtual Stochastic Costs is not convex in the departure time from the depot.

The second approach discussed by Lekkerkerker (2016) is Mixed Integer Programming (MIP). To get the optimum departure time, they model every simulation world with decision variables in the MIP. Using 100 simulation worlds the MIP requires 35 seconds to get the optimal departure time from the depot for a route with 15 orders. When increasing the number of orders on a route to 21, the MIP is not able to solve the problem to optimality within one minute. The optimization duration

Approach	Advantage	Disadvantage
Binary search using service reliabilities	Flexible	• Wrong objective function • Approximation
MIP	Exact	Slow
Backwards calculation	-	No driving regulations
Binary search using Virtual Stochastic Costs	Flexible	Approximation

Table 6.2: Overview of start time calculation methods

to solve the MIP fluctuates, but in general we can say that this MIP model is too slow to solve this problem.

The third approach by Lekkerkerker uses backwards calculation. For an order we can calculate for every simulation world at which time the vehicle should leave from the depot to serve the customer on time. Using the service requirement of every order we know in which simulation worlds we afford to be late, which gives us for every order a departure time from the depot such that the service requirement is exactly matched. This $O(n^2)$ algorithm calculates the minimum departure time for the different orders on a route. This approach works when using time-dependent travel times, but not when we are concerned with driving regulations. We cannot calculate the travel time backwards, because this depends on the time a break is scheduled.

The first three rows in Table 6.2 show the advantages and disadvantages of the approaches proposed by Lekkerkerker. This shows that the three approaches cannot be used for the DTPP. We propose to use a binary search approach using the Virtual Stochastic Costs to determine the best departure time from the depot. The reason is that this is a flexible approach where we can include driving regulations. The approach is an approximation due to the non-convexity of the Virtual Stochastic Costs, but should give a departure time that results in a lower Virtual Stochastic Costs compared to using expected travel times.

In the available time of this research it proved to not be feasible implement this, but we believe that this approach can give good results. Therefore, we more research is required for Hypothesis 7.

7

Computational study

This chapter presents the problem instances and experiments we have done to evaluate our proposed optimization algorithms. In Section 7.1 we present the different problem instances we have used in our experiments. Section 7.2 presents a set of additional hypotheses which were not covered by the hypotheses in the Introduction (Section 1.4). The experiments and the results are presented in Section 7.3.

7.1 Problem instances

To the best of our knowledge, there are no benchmarks for the DAIPEX Transportation Planning Problem or Vehicle Routing Problem with Stochastic Travel Times based on data from transportation companies and real-life travel data. Therefore, we have constructed a set of problem instances based on real-life data. We use instance data from a transportation company, hereafter referred to as company V. As discussed in Chapter 4, we use travel time data coming from data aggregation algorithms as developed by the IE&IS group of Eindhoven University of Technology which obtained travel time data from TomTom. We use 100 simulation worlds to evaluate the results.

The instance data from company V is based on Friday 18 September 2015, which was a regular day for transportation companies. On this day company V had to transport 357 orders to 318 different locations. We call this problem instance *V-357*. Based on *V-357* we have constructed a couple of other problem instances with different properties.

V-357 The base instance has 357 orders for 318 locations and the time windows are provided by company V. This instance has a service reliability requirement of 95% and a driving limit reliability requirement of 95%.

V-357 tight-50 As discussed in Section 1.1 the delivery business is concerned with time-boxed deliveries. There are time-boxes communicated by transportation companies to their customers and time-boxes required by customers because of unavailability outside of the time-boxes due to closed shops or closed city centers. Therefore, we have constructed a problem instance were we have tightened the time windows. Starting from problem instance *V-357* we have randomly selected 50% of the orders and tightened their time windows to 2 hours. When tightening a time window $[r_o, d_o]$ the new release date r_o is a random hour within the old time window. If the original time window is smaller than the required two hours, we do not change the time window. Using this approach 45% of the orders got a smaller time window, such that 50% of the orders have a time window of at most two hours.

V-357 service Orders from customers have different levels of importance. Some customers have orders which they would like to receive or pickup somewhere during the week, but they are indifferent about the moment of service. Other customers have orders which should not be received late under any circumstances because of economical or health reasons. Moreover, some time windows are strict where service is not possible outside of the time window, for example because of closed city centers or shops. Other time windows are flexible: the customer prefers to be served within the time window, but accepts being served outside the time window.

The importance of service within the time window can be defined in a contract with the customer. A transportation company can for example have a default service reliability requirement of 95%. If a customer agrees that a service reliability requirement of 75% is sufficient for them, the company can offer their services at a lower price. The contrary is also possible: demanding a service reliability requirement of 99% can be offered at a higher price.

Therefore, we constructed the *V-357 service* instance where 40% of the orders have a service reliability requirement of 75%, 50% of the orders requires 95% and the remaining 10% has a service reliability requirement of 99%. We use the *V-357 tight-50* as base instance, meaning that 50% of the orders have a time window of 2 hours or less.

V-357 driving Violating the daily driving limit can have the consequence of being fined or getting a driving ban. There are various options when faced with exceeding the driving limit:

- Continue driving and risk a fine.
- Send out another driver. Potential problem is that the original driver is still working, possibly exceeding the working limit.
- Eliminate some stops in order to be back at the depot before violating the driving limit. Not delivering all orders incurs penalty costs.
- Switch to a two day route. This is possible if a sleeping cabin is present in the truck.

Which of these options is best depends on the situation. What the options have in common is that they all incur additional costs. Therefore, considering this while making a transportation plan can save money. We have constructed the *V-357 driving* instance which has an increased driving limit reliability

Test instance	Time window	Service requirement	Driving limit requirement
V-357	Normal	95%	95%
V-357 tight-50	50% orders 2 hours time window	95%	95%
V-357 service	50% orders 2 hours time window	10% orders 99% 50% orders 95% 40% orders 75%	95%
V-357 driving	50% orders 2 hours time window	95%	99%

Table 7.1: Problem instances

requirement of 99%. The instance is based on *V-357 tight-50*, meaning that 50% of the orders have a time window of 2 hours.

The four problem instances used to evaluate our optimization algorithms are summarized in Table 7.1.

The cost structure includes a cost per hour and a cost per distance. There are no costs for resource usage, because they are included in the cost per hour and cost per distance. The costs per hour and cost per distance are not reported in this thesis because of confidentiality reasons. The Total Costs, Virtual Stochastic Costs and Virtual Expected Costs reported in this thesis are not the real costs, but a factor has been applied because of confidentiality reasons.

In our experiments we assume there is no limit on the number of transportation resources we can use. Moreover, we assume we have a homogeneous fleet of trucks and trailers. Every resource should make a single round trip: it is not allowed to load or unload at the depot apart from the end of the route. If in the resulting transportation plan the first resource finishes before the second resource starts, these routes can be executed by the same resource. This has the same costs, since we assume there is fixed costs for resource usage.

We evaluate every solution using the *Virtual Stochastic Costs* as defined in Section 3.4.4. We use the expected travel time to calculate the costs for the vehicle and the driver and we use the Sample Average Approximation approach to calculate the constraint costs. The reason is that we have seen that the costs using expected travel times are close to the costs based on the Sample Average Approximation (SAA) approach, while it requires more computation time to use the SAA approach. During the execution of the optimization algorithm multiple solution approaches use a different cost function: the *Virtual Expected Costs.* The difference is explained in Section 5.3.3.

7.2 Hypotheses

In Chapter 1 we have stated a set of hypotheses which we had before starting the project. Next to these hypotheses we have defined new hypotheses while conducting research.

Hypothesis 7 (Start time calculation)
Calculating departure times from the depot using expected travel times leads to orders with unmet service reliability requirement. Using travel time distributions to calculate the departure time can increase their service reliabilities.

Hypothesis 8 (Driving limit expected travel times)
Using the expected travel times results in many routes which violate the driving limit when evaluating using Sample Average Approximation.

Hypothesis 9 (Driving limit Sample Average Approximation)
Doing Sample Average Approximation works better than using the expected travel times to evaluate the driving limit.

Hypothesis 10 (Quick result)
When a plan is required within five minutes, using travel time slack performs better than using the travel time distributions in the optimization algorithm.

7.3 Experiments and results

For each of the four problem instances we have a solution with lowest Virtual Stochastic Costs. This has been found by running the travel time distribution optimizer multiple times for a long time until the optimization algorithm has converged every time. We can see the optimization algorithm has converged, because for example for the *V-357* instance in the last 3 hours no improvement has been found. Manual inspection of the solution neither gives any improvement. This does not mean the found solutions are the best possible solutions. The optimization algorithm uses Large Neighborhood Search which means the solution can be a local optimum. Nevertheless, we believe our best known solutions have a good Virtual Stochastic Costs, because of repeatedly finding solutions which are close to this best solution.

We use the terminology unreliable order and unreliable route in this chapter. An unreliable order refers to an order which does not satisfy the service reliability requirement. An unreliable route refers to a route which does not satisfy the driving limit reliability requirement.

Typically most information on orders is available at the evening of the day before the execution. The time available to generate a good transportation plan for company *V* is 60 minutes. Moreover, we use 60 minutes to get a fair comparison between the different solution approaches as discussed in Section 7.3.10.

The performed experiments are done using two different types of virtual machine from Quintiq. The first type is an Intel(R) Xeon(R) CPU E5-2660 v3 @2.60 GHz and 64GB RAM. The second type is an Intel(R) Xeon(R) CPU E5-2650 @ 2.00 GHz and 32GB RAM. There is a difference in hardware which we should compensate for by changing the allowed runtime. We have done equal experiments on both machines and compared the number of iterations. This showed that using the 64GB machine we could do 87% more iterations in an equal amount of time. Therefore, all results reported on in this chapter are based on 60 minutes runtime on the 64GB machine or an equivalent of $60 \cdot 1.87 = 112$ minutes on the 32GB machine.

The optimization algorithm used for the experiments is a random algorithm. Therefore, we have repeated a selected number of experiments multiple times to exclude the possibility of finding a result much better than the average. Time did not allow us to repeat every experiment multiple times, but comparing the results from different settings we can see that we cannot expect to get much better results with most settings. When performing experiments with multiple settings for the same solution approach we have first done a single run for every setting. We have repeated the three settings which give best results based on the single run three times. For every experiment we indicate if this is the result of a single run or the average of multiple runs. If this is the result of multiple runs, we report on average values which means we can expect rational numbers for routes and orders. We refer to Appendix B for detailed results for the different experiments.

The experiments reported on in this chapter are conducted using a single time period. Using multiple time periods significantly decreases the performance of the optimization algorithm. We have selected the time period with the largest variance in the travel times, which is the morning rush, to use for the full day because this is the most difficult time period. The solution obtained by solving the DTPP for the time period with the largest variance can also be used for the other time periods which have less variance in the travel times.

7.3.1 Using expected travel time

Hypothesis 1 states that using expected travel times when calculating a transportation plan gives bad plans in terms of service reliability. Table 7.2 shows the number of unreliable orders and the largest service reliability requirement violation for the four problem instances using the expected travel time. *V-357* has the normal time windows while for the other three problem instances at least 50% of the orders have tightened time windows. We see that 23% of the 357 orders in the *V-357* problem instance are not serviced with their required service reliability requirement. This increasing to 30% to 39% for the other three problem instances which have tighter time windows. Table 7.2 also shows that the largest violation of the service reliability requirement is to 0.64, which means this order is serviced with a reliability of $0.95 - 0.64 = 0.31$.

The bad plans are reflected by the Virtual Stochastic Costs as shown in Table 7.2. The optimization algorithm does not take the service reliability requirement into

Problem instance	Virtual Stochastic Costs			#unreliable orders	Largest violation unreliable order
	Exp.	Best	Δ		$0 \leq viol. \leq 1$
V-357	37,462	12,896	190%	83 (23%)	0.44
V-357 tight-50	56,822	17,211	230%	134 (38%)	0.55
V-357 service	46,896	16,575	183%	106 (30%)	0.52
V-357 driving	59,704	21,674	175%	140 (39%)	0.64

Table 7.2: Using the expected travel time (shown in column Exp.) gives bad plans with many unreliable orders. The results are averages of three 1-hour runs. Δ gives how much worse the plan gets when using the expected travel time. The percentage for the #unreliable orders shows which percentage of the 357 orders are unreliable.

account which explains the high number of unreliable orders and supports Hypothesis 1.

7.3.2 Time window slack

Figure 7.1 shows the Virtual Stochastic Costs when increasing the time window slack for the four problem instances, as discussed in Section 5.2.1. There is a clear difference between the *V-357* problem instance and the three other problem instances. For *V-357* we can add time window slack to decrease the Virtual Stochastic Costs, but using more than 90 minutes time window slack does not gain much for the Virtual Stochastic Costs. We can see in Table B.2 that using more than 90 minutes time window slack gives solutions in which the number of unreliable orders decreases, but at the additional costs of more distance due to using more routes. For the problem instances *V-357 tight-50*, *V-357 service* and *V-357 driving* using 120 minutes time window slack gives the lowest Virtual Stochastic Costs. Adding more has the opposite effect: the solutions get worse scores. The reason for this is that the *V-357* problem instance does not have tight time windows while the other three have tight time windows. When arriving too early add a customer the driver has to wait and this waiting increases when adding more time window slack.

The solutions found with the best setting for time window slack are respectively 50%, 63%, 68%, and 60% worse than the best known result as shown in Tables B.2, B.5, B.8, and B.11.

7.3.3 Travel time slack using fixed percentage

Figure 7.2 presents the results when overestimating travel times using a fixed percentage. We see that the Virtual Stochastic Costs decrease as the fixed percentage increases. This only holds up to a certain amount of slack after which the additional costs for driving no longer outweighs the increased number of orders which

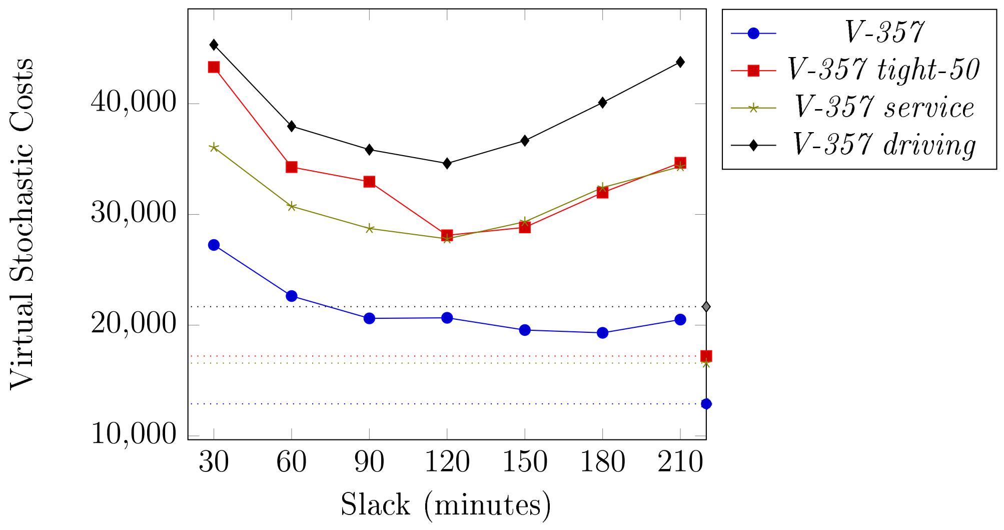


Figure 7.1: Time window slack in minutes. The dotted lines show the best known solution.

meet their service reliability requirement. The percentage to use for the travel time slack to minimize the Virtual Stochastic Costs is around 70%. We can see that the solutions obtained with overestimating the travel times using 90% have scores similar to the ones using 70%. The exact amount of slack depends on the problem instance and should be determined more accurately when applying this technique in practice. Finding the right amount of slack is difficult, because using too much slack generates transportation plan which uses many resources while reserving too little slack causes many unmet service reliabilities.

The best setting for travel time slack with a fixed percentage gives solutions of 26%, 27%, 35% and 23% worse results than the best known solutions as can be seen in Tables B.3, B.6, B.9, and B.12 in Appendix B. However, travel time slack using a fixed percentage gives 50% better results than using the expected travel time as can be seen in Figure 7.5.

Hypothesis 3 states that overestimating travel times using a fixed percentage results in either many unmet service reliability requirements or an inefficient use of resources. We see in Tables B.3, B.6, B.9, and B.12 in Appendix B that overestimating the travel times indeed results in using more resources and that to get a comparable number of unreliable orders more resources are needed than the best known solution.

7.3.4 Travel time slack based on Standard Deviation

Figure 7.3 shows the results of using travel time slack based on the standard deviation. As explained in Section 5.2 we overestimate the travel time on an arc by adding slack. We find the amount of slack by multiplying the standard deviation of the travel times in the simulation worlds with a factor. Preliminary tests have

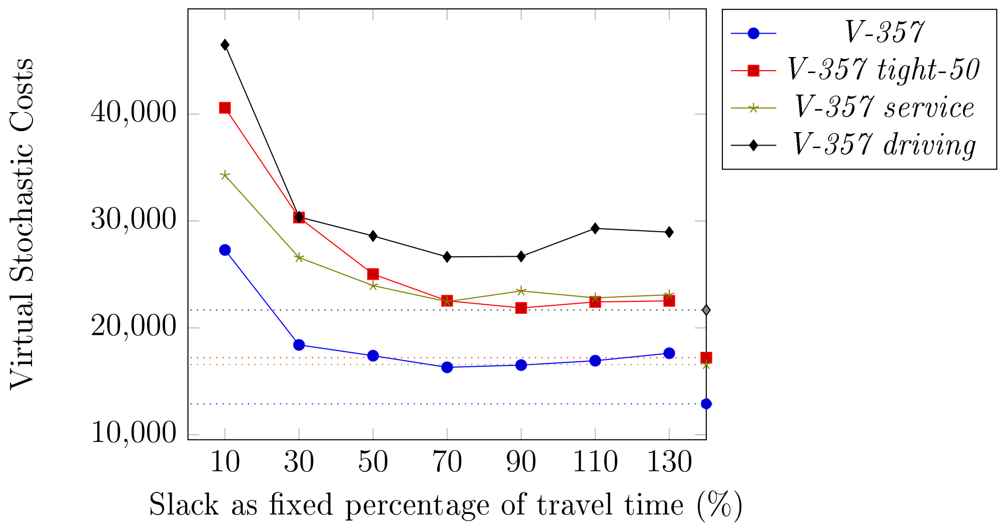


Figure 7.2: Travel time slack as a fixed percentage of the travel time. The dotted lines show the best known solution.

shown that the factor should be in the range of 0.5 to 5.5, which is also supported by the results.

We see that using travel time slack based on standard deviation can give much better results than using the expected travel time. We first conducted experiments using a factor from 0.5 to 5.5 with steps of 1.0. Figure 7.3 shows there is a large improvement when using 1.5 instead of 0.5 as factor for the travel time slack based on standard deviation. Therefore, we included an extra experiment with a factor of 1.0. For problem instance *V-357* the Virtual Stochastic Costs is almost equal compared to using a factor of 1.5 (0.2% difference which can be contributed to randomness). For the other three instances the Virtual Stochastic Costs gets worse when decreasing the factor to 1.0. For this reason, a factor of 1.5 should be used for the travel time slack based on standard deviation which is independent of our four problem instance. Adding more slack quickly gives worse results on all four problem instances.

Using a factor of 1.5 for the slack based on standard deviation gives respectively 18%, 17%, 21% and 17% worse results than the best known solution. Hypothesis 4, which states that overestimating the travel time based on standard deviation of travel times works better than adding a fixed percentage to the travel time, is supported.

Table B.3 in Appendix B shows the additional results of the experiments with travel time slack based on standard deviation. While using the expected travel times gives a transportation plan with 83 unreliable orders, this decreases to 39 when using a factor of 0.5, to 9 when using 1.0 and 3.7 when using 1.5. We can even generate a transportation plan where all orders are served with the service reliability requirement: using a factor of at least 2.5 for the travel time slack based on standard deviation gives plans in which no order is unreliable. This is at the expense of increasing costs. The best solution without any unreliable order has Virtual Stochastic Costs of 16,646, while the best known solution has a score of 12,896 but with one

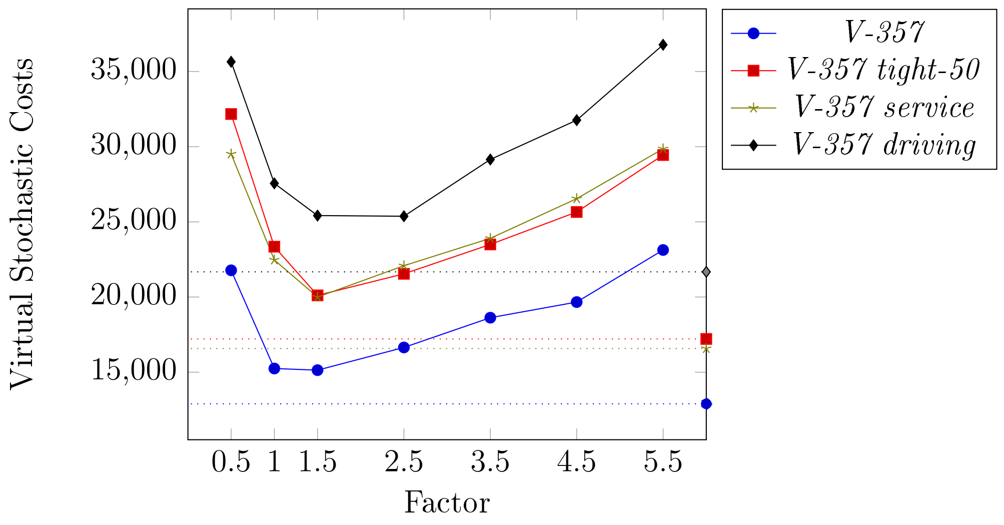


Figure 7.3: Travel time slack based on standard deviation. The dotted lines show the best known solution.

unreliable order. This shows that the optimization algorithm does not only focus on the number of unreliable orders, but rather compares solutions based on the Virtual Stochastic Costs. One can use different parameters to change the weight of service reliability requirement. This optimization algorithm can thus give solutions with different trade-off between costs and service reliabilities.

Related to the standard deviation is the variance: $variance = \sigma^2$. One might ask why we have not tested using travel time slack based on the variance. The first reason is that the variance is in a different order of magnitude than the standard deviation, and the standard deviation is in the same magnitude as travel times. The second reason is the size of the variance. If the standard deviation of the travel time on an arc is less than 1.0, then the variance is even smaller due to the square. Using the travel times in hours gives a standard deviation which is often less than 1.0, resulting in a small variance. We could have done the computation for the variance in minutes to account for this, but then we are still faced with the problem of a different order of magnitude. Therefore, we have not included results on using travel time slack based on the variance.

7.3.5 Travel time slack based on MAD

Solutions obtained with overestimating travel times based on MAD have lower Virtual Stochastic Costs than when using the expected travel time, as can be seen in Tables 7.3, 7.4, 7.5, and 7.6. Figure 7.4 show the results of using travel time slack based on MAD for the four problem instances. While for the travel time slack based on standard deviation the Virtual Stochastic Costs quickly increases when adding too much slack, this is not the case when using the MAD. This is a strong point in favor of using travel time slack based on MAD. This optimization algorithm is less sensitive for the factor and therefore more likely to do well on a range of problem

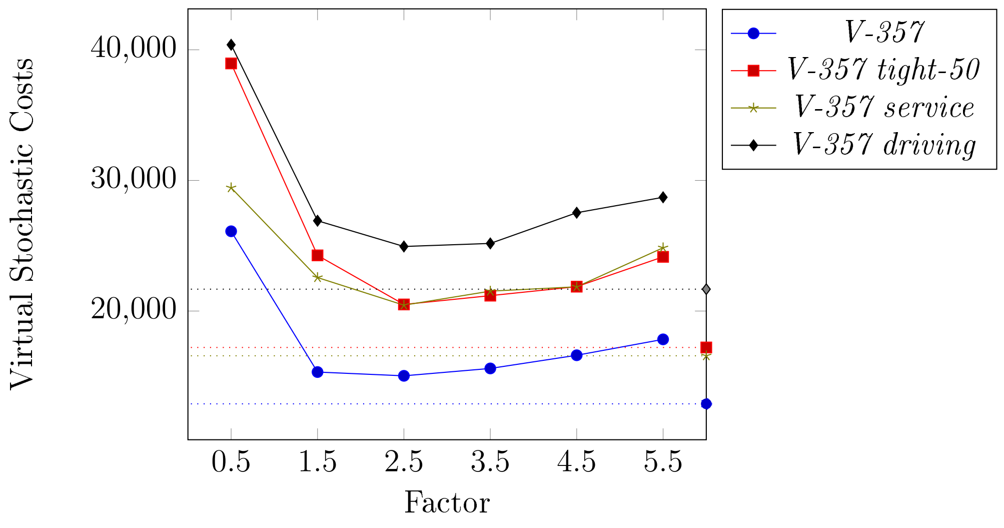


Figure 7.4: Travel time slack based on Mean Absolute Deviation from the mean (MAD). The dotted lines show the best known solution.

instances.

We see that using a factor of 2.5 for the MAD based slack gives the best results, but when using a factor of 1.5 or 3.5 the difference in Virtual Stochastic Costs is relatively small.

Hypothesis 5 states that building in slack by overestimating the travel times based on the Mean Absolute Deviation from the Mean (MAD) works better than adding a fixed percentage to the travel time. Using travel time slack based on MAD gives solutions which are respectively 17%, 19%, 23%, and 15% worse than the best known results. For every problem instance this is better than using a fixed percentage for the travel time slack, which supports Hypothesis 5.

Similar as when using travel time slack based on standard deviation, we can also obtain transportation plans without unreliable orders by increasing the factor for the MAD. This is again at the expense of more routes and thus a higher costs and more distance, giving a higher Virtual Stochastic Costs.

7.3.6 Sample Average Approximation

Hypothesis 6 states that using Sample Average Approximation (SAA) works best in terms of quality, but takes more time and requires much more data of much higher quality. The latter part is supported by Chapter 4 where we have explained which data is needed for the SAA approach. Section 7.3.10 discusses that doing SAA takes more time than using deterministic travel times. This section shows that the SAA approach works best in terms of quality.

Figure 7.5 shows the result of all solution approaches to the four problem instances where the best parameter setting is used for the different types of slack. The best known solutions are obtained repeatedly running the optimization algorithm using

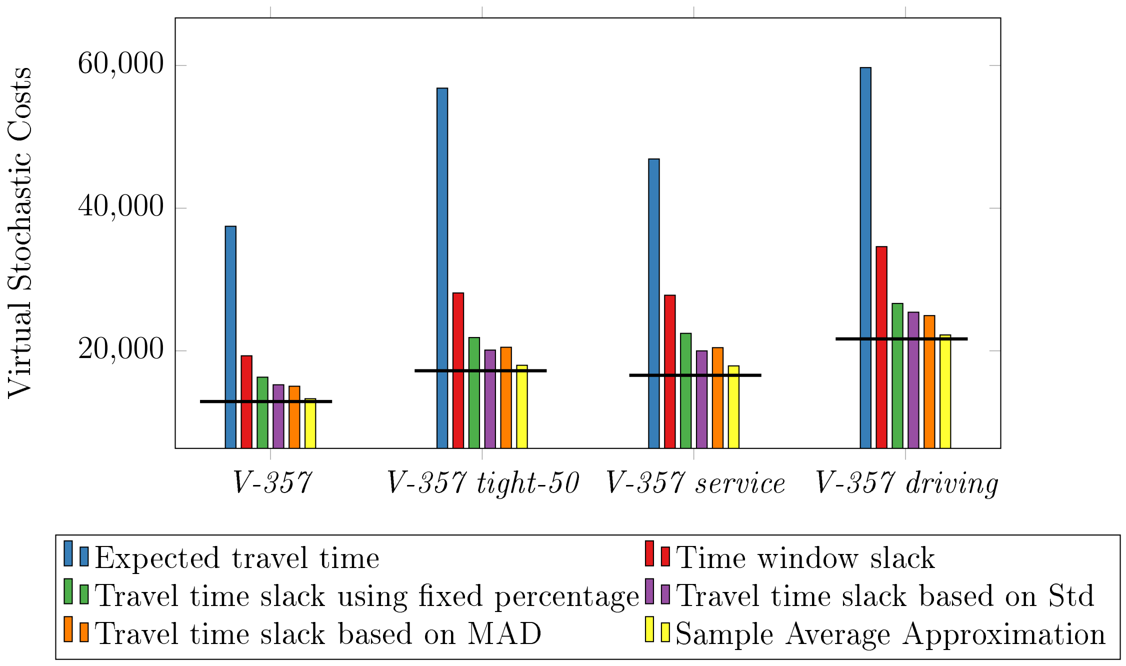


Figure 7.5: Comparison of all solution approaches with for every solution approach the best parameter setting. Horizontal lines show best known solutions.

Sample Average Approximation for a long period. We can see that using Sample Average Approximation performs best of the different solution approaches for the four problem instances.

Tables 7.3, 7.4, 7.5 and 7.6 show the Virtual Stochastic Cost for all solution approaches using the best parameter setting for the problem instances *V-357*, *V-357 tight-50*, *V-357 service* and *V-357 driving*. We can see that for problem instances *V-357* and *V-357 driving* the Sample Average Approximation gives solutions which are only 3% worse than the best known solution. For problem instances *V-357 tight-50* and *V-357 service* the solutions are respectively 4 and 8% away from the best known solution.

7.3.7 Comparison solution approaches

We have seen in Section 7.3.1 that using expected travel times gives bad plans in terms of service reliability requirements. When using time window slack the solutions are better than the solutions obtained with expected travel times, but they are still around 50% worse than the best known solutions. Therefore, we focus our comparison on the remaining solution approaches: using different types of travel time slack and using Sample Average Approximation in the optimization algorithm.

Figure 7.6 shows the four solution approaches with the best results together with the best known solution, filtered from Figure 7.5 to get a better comparison of the approaches. We see that using travel time slack with a fixed percentage obtains

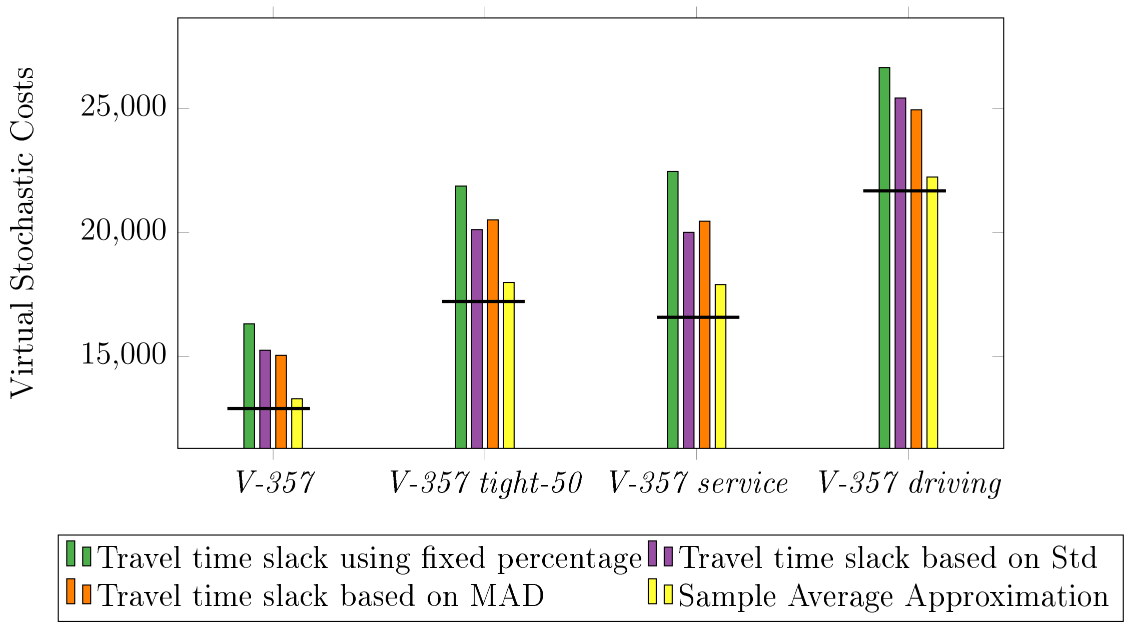


Figure 7.6: Comparison of best four solution approaches (excluding expected travel time and time window slack) with for every solution approach the best parameter setting.

worse results on the four problem instances than the other three solution approaches. There is not much difference between using travel time slack based on standard deviation or based on MAD when one compare the best results. For *V-357 tight-50* and *V-357 service* MAD performs slightly better while for *V-357 driving* using the standard deviation gives slightly better results. Using travel time slack based on MAD is more robust with relation to the chosen factor. Doing Sample Average Approximation gives clearly the best results on all four problem instances. How much the solution is away from the best known solution depends on the problem instance. As said, while this is only 3% for *V-357* and *V-357 driving*, it is respectively 4% and 8% away for *V-357 tight-50* and *V-357 service*.

Hypothesis 2 states that using travel time slack works better than using time window slack. The results in Tables 7.3, 7.4, 7.5, and 7.6 indeed show that all thee types of travel time slack work better than using time window slack.

7.3.8 Driving limit

Hypothesis 8 states that using expected travel times results in many routes which violate the driving limit reliability requirement and Hypothesis 9 states that doing Sample Average Approximations works better than using expected travel times. We discuss these hypotheses in the next sections.

Comparing the number of unreliable routes does not give us all relevant information if driving limit violations get solved when using slack. The reason is that we can have two transportation plans with the size of the violations very different. For example,

	V-357	
Solution approach	Virtual Stochastic Costs	Δ
Expected travel time	37,462	190%
Time window slack: 180 min.	19,309	50%
Travel time slack fixed percentage: 70%	16,312	26%
Travel time slack Std * 1.0	15,245	18%
Travel time slack MAD * 2.5	15,041	17%
Sample Average Approximation	13,291	3%
Best known solution	12,896	0%

Table 7.3: Best setting for each solution approach for problem instance *V-357*.

	V-357 tight-50	
Solution approach	Virtual Stochastic Costs	Δ
Expected travel time	56,822	230%
Time window slack: 120 min.	28,117	63%
Travel time slack fixed percentage: 90%	21,866	27%
Travel time slack Std * 1.5	20,110	17%
Travel time slack MAD * 2.5	20,504	19%
Sample Average Approximation	17,977	4%
Best known solution	17,211	0%

Table 7.4: Best setting for each solution approach for problem instance *V-357 tight-50*

	V-357 service	
Solution approach	Virtual Stochastic Costs	Δ
Expected travel time	46,896	183%
Time window slack: 120 min.	27,798	68%
Travel time slack fixed percentage: 70%	22,454	35%
Travel time slack Std * 1.5	19,223	16%
Travel time slack MAD * 2.5	20,449	23%
Sample Average Approximation	17,893	8%
Best known solution	16,575	0%

Table 7.5: Best setting for each solution approach for problem instance *V-357 service*

	V-357 driving	
Solution approach	Virtual Stochastic Costs	Δ
Expected travel time	59,704	175%
Time window slack: 120 min.	34,610	60%
Travel time slack fixed percentage: 70%	26,643	23%
Travel time slack Std * 1.5	25,417	17%
Travel time slack MAD * 2.5	24,942	15%
Sample Average Approximation	22,231	3%
Best known solution	21,674	0%

Table 7.6: Best setting for each solution approach for problem instance *V-357 driving*

the first plan has 10 unreliable routes and every unreliable route has a violation of 0.1, giving a sum of violation of 1. The second plan also has 10 unreliable routes, but these unreliable routes all have a violation of 0.3, giving a sum of violation of 3. The first plan is clearly better in terms of the driving limit reliability requirement. Therefore, we compare the sum of the violations of the unreliable routes.

7.3.8.1 Expected travel time

In Table B.1 in Appendix B we indeed see that 9 out of the 22.3 routes do not satisfy the driving limit reliability requirement (there are 22.3 routes, because we report the average of three runs).

We look more closely to one of the three transportation plans which has 9 unreliable routes out of the 23 routes. In this transportation plan there are two routes with a driving time of respectively 8:55 hours and 8:58 hours. Only a small disturbance in the travel time compared to the expected travel time results in a violation of the 9-hour driving limit. The two routes have a driving limit reliability of respectively 0.75 and 0.60, which means they have a violation of 0.20 and 0.35. Next to these two long routes, there are five routes with expected driving duration of between 8:10 hours and 8:30 hours. These routes have varying driving limit reliabilities of 0.70 to 0.87, but they all have a violation. The sum of all driving limit violations is 1.26. This supports Hypothesis 8 that using expected travel times results in many routes which violate the driving limit reliability requirement.

7.3.8.2 Using slack

We showed in Section 5.2.1 that we use an adapted driving limit when using time window slack. In the left figure in Figure 7.7 we see that using time window slack the sum of violations decreases when increasing the slack. This gives a reduction from 1.26 for no time window slack to 0.58 when using 60 minutes time window slack. Increasing the amount of slack does not decrease the sum of violations much.

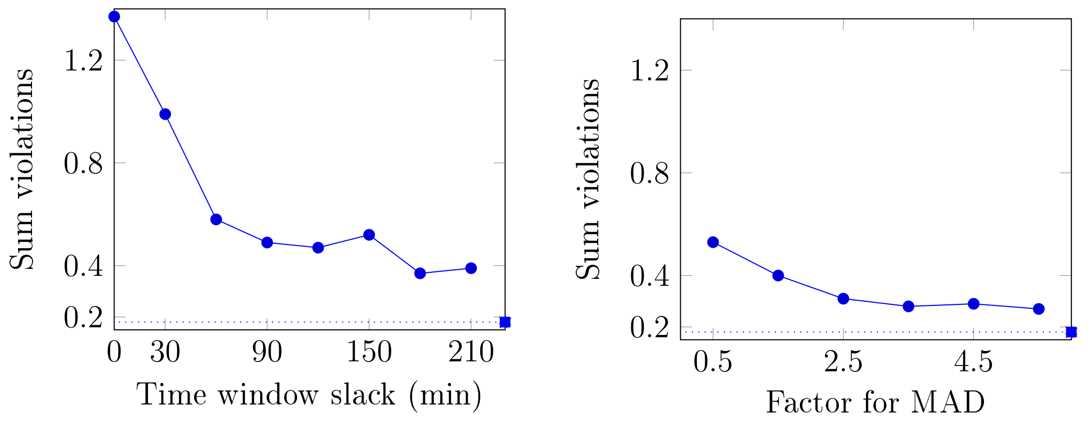


Figure 7.7: Sum of driving limit violations for *V-357* using time window slack and travel time slack based on MAD. The dotted lines show th sum of violations for the best known solution.

This shows that time window slack is not no good solution to meet the driving limit requirements.

We show in the right figure in Figure 7.7 how travel time slack performs with relation to the driving limit reliability. We see that this gives better results than using time window slack. The sum of violations decreases when increasing the factor for the travel time slack. The number of unreliable routes decreases from 9 when using expected travel times to 4 when using travel time slack based on MAD with a factor of 4.5. Increasing the factor more could decrease the sum of violations, but more experiments should confirm this. This can be explained, because using a higher factor results in longer travel times such that less orders are placed on a route. We then more likely satisfy the driving limit reliability requirement.

7.3.8.3 Sample Average Approximation

Hypothesis 9 states that doing Sample Average Approximation (SAA) works better than using the expected travel time to evaluate the driving limit. In the previous section we have already seen that using slack can only partly alleviate the driving limit violations when using the expected travel time.

When doing SAA the number of unreliable routes is reduced to two and they have 0.18 violation together. Both the number of routes as the sum of the violation is lower than using expected travel times or using slack. It is an improvement of 86% on the sum of the violations, which we have used to compare the transportation plans.

In the transportation plan obtained with SAA, the two routes with the violation have an expected driving time of 7:51 and 8:22 hours and we see that all other routes have an expected travel time of at most 7:30 hours. We see that those two routes visit customers far from the depot. Ensuring the driving limit requirement is met, can be done by using an extra route. Distributing the orders over the two routes is in general not a good idea: although we do not have a cost for resource usage, this vehicle has to drive far incurring high costs. This means that if there are multiple orders which are far from the depot, but those orders are together, then using a single route for all orders thereby likely violating the driving limit requirement can be the best solution. If transportation companies belief differently, then the costs for violating the driving limit reliability should be increased.

7.3.9 Lowered reliability costs

In the previous experiments we have always assumed the goal was to satisfy all service reliability requirements and driving limit reliability requirements. We have seen the costs are higher when satisfying the reliability requirements compared to using expected travel times where there are many violations. We can also decide to make the reliability less important and focusing more to get a short total distance and low total costs.

We can do this by changing the factor in the penalty cost function as defined in Section 3.4. We have a factor f_s for the service reliability requirement and a factor f_dl for the driving limit reliability requirement. They have default values of respectively 200 and 400. In the experiments in this section we use a factor of 50 for the driving limit and we vary the factor for the service requirement.

Table 7.7 shows the experiments with changed factors for the service requirement. We see that lowering the factor for the service reliability, gives transportation plans with increasing number of unreliable orders. More unreliable orders allows to decrease the costs and the distance, as can be seen in the table. We see that using a factor lower than 2 gives solutions with very similar costs, while the number of unreliable orders differs much (from 47 to 81). This means constraint costs for the service reliability requirement is very low compared to the costs of the transportation plan.

7.3.10 Convergence of optimization algorithms

The optimization algorithms used in this research are based on Large Neighborhood Search (LNS). These algorithms are based on randomness and on doing many iterations. After doing a certain amount of iterations, the LNS is in a local (or global) optimum. Continuing the search does not much gain in this case; only little improvements will be found when doing many iterations. Before arriving in a local optimum, most iterations give an improvement. Therefore, we should verify if the optimization algorithm has converged, because otherwise we could get a better solution by running the optimization for a longer period.

Figure 7.8 shows the objective value during the search for two optimization runs. In Figure 7.8a we used travel time slack based on standard deviation. We should verify the convergence using the Virtual Expected Costs, because that is the objective which is used in the optimization algorithm. After 9 minutes we obtained a solution

#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Service reliability factor f_s
30	11,365	7,320	1	0.29	2	0.09	0.18	$f_s = 200$
28	10,700	6,720	5	0.29	4	0.1	0.33	$f_s = 20$
28	10,590	6,495	21	0.95	4	0.21	0.53	$f_s = 10$
28	10,412	6,336	31	0.95	5	0.19	0.67	$f_s = 5$
24	9,715	5,806	47	0.95	5	0.21	0.7	$f_s = 2$
24	9,714	5,757	61	0.95	6	0.15	0.54	$f_s = 1$
26	9,726	5,821	81	0.95	6	0.37	1.07	$f_s = 0.5$

Table 7.7: Problem instance *V-357* using lowered service reliability costs. Factor f_dl for the driving limit reliability requirement is 50. Sample Average Approximation approach is used.

with score within 1% of the final solution and we see that there is hardly any improvement after 20 minutes. For these reasons, we can conclude the optimization algorithm has converged. This not only applies when using travel time slack with standard deviation, but in general when using deterministic travel times.

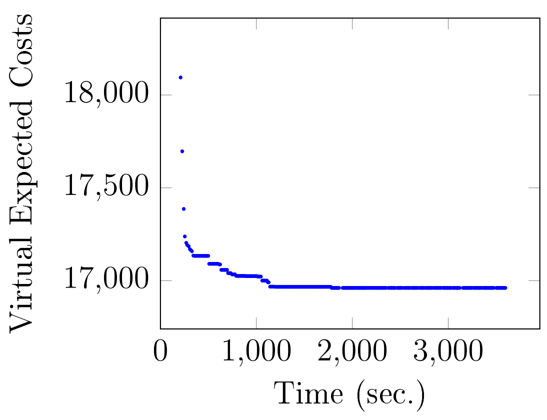
It is a different story when doing Sample Average Approximation (SAA). We verify the convergence using the Virtual Stochastic Costs in this case. We can see in Figure 7.8b that even after 50 minutes the optimization algorithm still finds improvements. To get a fair comparison, we have limited the optimization duration to 60 minutes, but increasing the optimization duration should allow to find more improvements when doing SAA. We have already made improvements to the LNS to converge faster, but we believe there is still room for improvement.

We have a potential improvement which is not yet supported by POA. POA initializes the subproblem at the start of every iteration. This takes a significant part of the optimization duration due to the initialization of the simulation words. The travel times in these simulation worlds do not change, so initializing this once at the start of the optimization and then reusing them at every iteration can improve the speed of the optimizers up to 20% we believe.

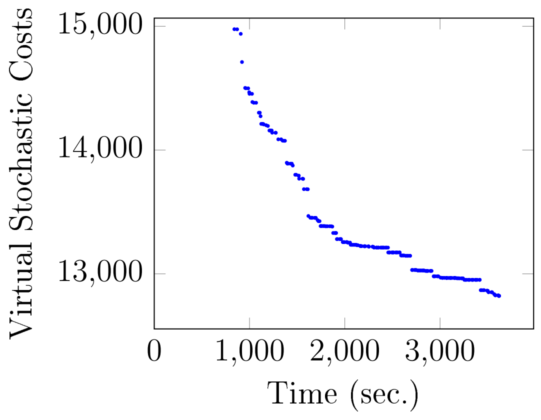
Hypothesis 6 states among others that Sample Average Approximation takes more time. Looking to the convergence results of the two optimization algorithms, we conclude that the hypothesis is supported.

7.3.11 Limited number of resources

In the previous sections we assumed that the number of vehicles is unlimited. One consequence is that the number of vehicles used in a transportation plan can be high, in particular when using much slack. Table B.1 in Appendix B shows that when



(a) Using travel time slack based on standard deviation with factor 1.5



(b) Using Sample Average Approximation

Figure 7.8: Convergence of two optimization algorithms on *V-357*.

using expected travel times we obtain a transportation plan with 22.3 routes. The best known solution obtained using Sample Average Approximation uses 30 routes and is 190% better than the solution obtained using expected travel times.

If we limit the number of resources when using Sample Average Approximation (SAA), we should still be able to find a solution at least as good as the solution from the expected travel times. Exactly this solution is still valid for the SAA, but we are interested if it can also be found. Therefore, we have done several experiments on *V-357* using SAA with limited number of routes. While conducting the experiments we have seen that 60 minutes optimization duration is not enough when limiting the number of routes. To make sure the optimization algorithm is finished, we have used 8 hours for these runs.

Figure 7.9 shows the Virtual Stochastic Costs when the number of routes is limited on problem instance *V-357* using the SAA. We have also included the Virtual Stochastic Costs when using expected travel times. We see that using 19 routes is not enough: the Virtual Stochastic Costs is very high due to 47 unreliable orders and 8 unreliable routes. Increasing the number of available routes decreases the Virtual Stochastic Costs, but only up to 29 routes because then the objective remains equal.

The solution from the expected travel time uses 22 or 23 routes. We selected the solution with 23 routes and conducted an experiment using Sample Average Approximation with 23 routes resources available. Table 7.8 shows the results of using expected travel times and using Sample Average Approximation with 23. We see that a better solution is obtained using SAA compared to using expected travel times in terms of Virtual Stochastic Costs. We see we can decrease the number of unreliable orders from 83 to 12, but at the expend of increasing costs.

This shows that even if a transportation cannot afford to buy an additional resource, using the existing resources we can still generate better transportation plans than using expected travel times. Increasing the number of resources allows to gain even

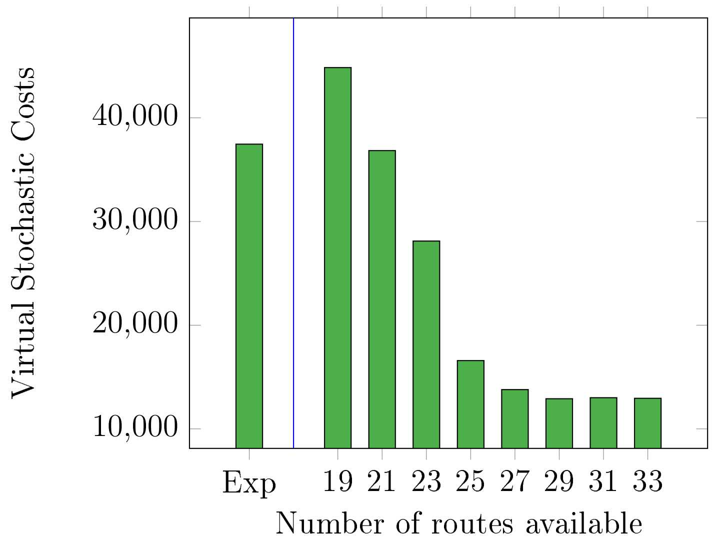


Figure 7.9: Limited number of routes using Sample Average Approximation on *V-357* and using expected travel times on unlimited number of routes.

Solution approach	#R	Virtual Stochastic Costs	Costs	Dist.	#unrel. orders	#unrel. routes	Sum violat. unrel. routes
Expected travel times	23	35,432	9,733	5,856	83	9	1.26
Sample Average Approximation 23 routes	23	28,116	10,645	6,571	12	6	1.44

Table 7.8: Sample Average Approximation using 23 routes, because solution expected travel times uses 23 routes.

more in Virtual Stochastic Costs and transportation companies can make a trade-off between these two.

7.3.12 Quick result

Hypothesis 6 states that doing Sample Average Approximation takes more time than using time window slack or travel time slack. This is formalized in Hypothesis 10 by stating that if a transportation plan is required within 5 minutes, then travel time slack outperforms Sample Average Approximation. We tested the hypothesis on problem instance *V-357* using a 5 minute optimization duration.

Figure 7.10 provides support for Hypothesis 10 which gives averages for three experiments. Although doing Sample Average Approximation performs better than using expected travel times, it performs much worse compared to using slack. Using time

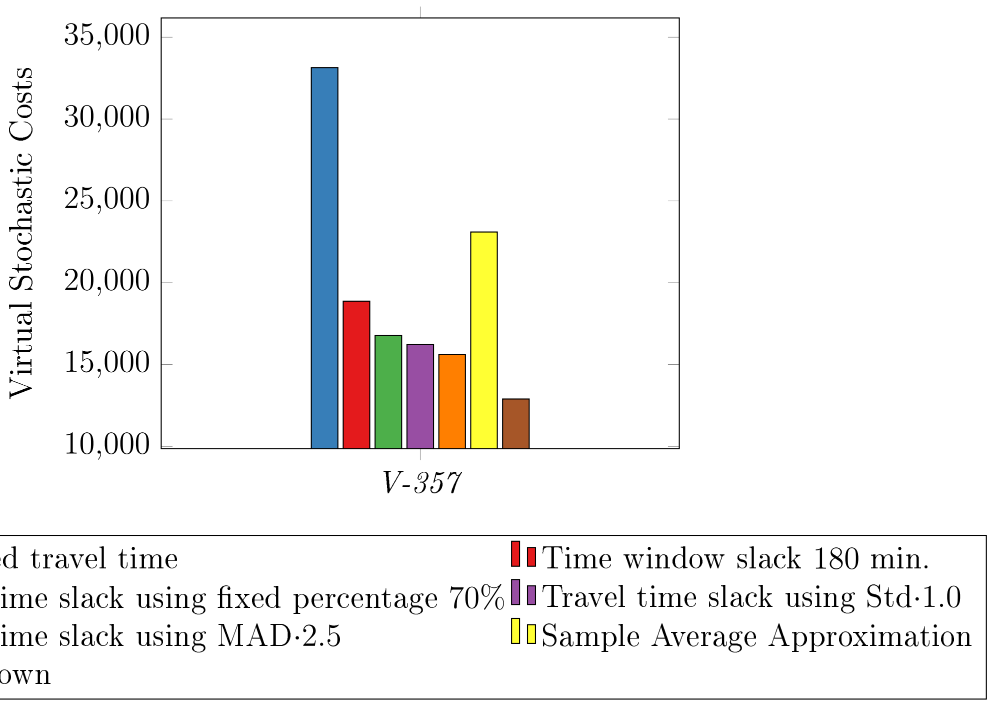


Figure 7.10: Results after 5 minutes optimization for problem instance *V-357* using three runs.

window slack, which performs badly when enough optimization time is available, now performs better than using Sample Average Approximation. Using travel time slack performs even better than using time window slack and which type of travel time slack does not matter much.

Looking to the individual results from every solution approach shows large variety. For example, the three experiments when doing Sample Average Approximation give Virtual Stochastic Costs of 24,896, 18,735, and 25,672. This shows that the optimizer is far from converged after 5 minutes.

What we also see is that a transportation plan can become worse when increasing the optimization duration when using slack. The reason is that when using slack, the optimizer uses the Virtual Expected Costs to evaluate intermediate solutions. This objective does not include the service reliabilities nor the driving limit reliabilities. As a result, the optimizer might change the transportation plan such that orders are served more closely towards the due date of its time windows. When evaluating using the Virtual Expected Costs it does not make a difference if the order is served 1 minute or 1 hour before the due date of the time window, but this matters much for the Virtual Stochastic Costs. As a result, it can happen that the best solution obtained when using slack is not the final solution and therefore, we do not report on this.

Hypothesis	Confirmed	More research required
Hypothesis 1	o	
Hypothesis 2	o	
Hypothesis 3	o	
Hypothesis 4	o	
Hypothesis 5	o	
Hypothesis 6	o	
Hypothesis 7		o
Hypothesis 8	o	
Hypothesis 9	o	
Hypothesis 10	o	

Table 7.9: Overview which hypotheses are confirmed, which ones are refuted and which require more research.

7.4 Conclusions

We have seen that using expected travel times gives bad transportation plans in terms of service reliabilities and driving limit reliabilities Using time window slack improves the transportation plans a lot, but is still around 70% worse than the best known solutions. Better solutions can be obtained using travel time slack.

We have seen that using travel time slack with a fixed percentage allows to obtain transportation plans which are better than those obtained using time window slack. However, these transportation plans are still around 30% worse than the best known solutions. This can be improved by using travel time slack based on the variance of the travel times. Using the standard deviation or the mean absolute deviation from the mean (MAD) allows to obtain better transportation plans. Depending on the problem instance this can be 20% worse than the best known solution. While using the standard deviation is very sensitive, using the MAD is less sensitive for the factor to scale the slack and thus more likely to do well on a range of problem instances.

The best transportation plans are obtained when using travel time distributions in the optimization algorithm. We have shown that using Sample Average Approximation allows to obtain transportation plans which are 3 to 8% worse than the best known solutions.

Table 7.9 shows which hypotheses are confirmed and which ones require more research. We can see that Hypothesis 7 requires more research. We discussed in Chapter 6 several methods to calculate the departure time from the depot. In the available time for this research it proved to not be feasible to implement these methods.

8

Conclusion and Future Work

This chapter presents the found conclusions of this thesis in Section 8.1 and discusses some ideas for future research in Section 8.2.

8.1 Conclusion

In this thesis we studied the DAIPEX Transportation Planning Problem (DTPP). The DTPP captures a broad array of real-life vehicle routing problems faced by transportation companies. One of the challenges faced are stochastic time-dependent travel times making transportation plans that are made in advance become suboptimal or even infeasible during execution. Another important issue are driving regulations including driving breaks and a daily driving limit. Furthermore, the DTPP includes time windows and cost structures as used by transportation companies.

We introduced a new way of evaluating the lateness of orders and the driving limit when considering stochastic travel times. The proposed measures allow to express with how much certainty an order should be served and with how much certainty the driving limit should be satisfied.

Generated transportation plans are evaluated on real-life instances using travel time data. This travel time data can come from TomTom or from logs coming from handheld devices to get travel time data for a company's own trucks. We introduced a data completion algorithm capable to calculate realistic travel time distributions which are originally missing. Using a hierarchical sampling mechanism we got realistic simulations taking geographical dependencies into account.

We have seen that using expected travel times when calculating transportation plans gives bad plans in terms of service reliabilities and driving limit reliabilities. To improve this we proposed several solution approaches for the DTPP. The first solution approach uses time window slack by using a tighter time window in the optimization algorithm. This improves the transportation plans, but cannot solve the many unmet service and driving limit reliabilities.

The second type of solution approach we introduced is travel time slack where we overestimate travel times. By means of this we obtain transportation plans that have more orders meet their service reliability requirement. An imprecise overestimation results in using too many vehicles which increases the distance and the costs of a transportation plan. We showed that overestimating travel times based on the standard deviation or on the mean absolute deviation from the mean works much better than using expected travel times or using time window slack.

Nevertheless, these solution approaches are still around 20% away from the best known solutions. These best known solutions are obtained by the optimization algorithm we developed that explicitly uses travel time distributions. This widely applicable algorithm outperforms all other solution approaches and allows to make a trade-off between logistics costs and robustness of a transportation plan.

8.2 Future Work

While conducting the research we came up with several potential directions for future research.

Trajectory method Looking at space-time diagrams we can perform a realistic simulation of the route duration for a specific historical departure time by using the Trajectory Method as introduced by Lint and Zjipp (2003). This requires much data, since we need for each simulation world one day in history with enough available data, but it more accurately represents the real world.

Satisfying triangle inequality Our hierarchical sampling method does not enforce the triangle inequality. This can be fixed in different ways, for example by solving the metric nearness problem which minimizes distance between distance matrices. One could use sampling where we verify for each new sample if the triangle inequality is still satisfied and resample otherwise. Many iterations can be necessary and the average travel time in the resulting simulations might be different than the expected travel time.

POA MultiInTimeConstraint In Section 5.3.2.2 we explained that we use the EndConstraint and corresponding MultiEndConstraint of POA to verify the lateness of orders. We explained that we prefer to use the InTimeConstraint and the corresponding MultiInTimeConstraint, but the latter is not supported by POA. We advice Quintiq to make the MultiInTimeConstraint available in POA.

Stochastic and time-dependent service times We have focused on fixed service times, but in real-life service times are stochastic. Time dependent service times could be needed when locations have busy hours in which servicing takes longer than usual. Implementing time dependency in the simulations in the model is possible by generating a time dependent sample matrix. For example one probability distribution is used when the service takes place in the time window of the order and another probability distribution is used when the service takes place outside of the time window. When evaluating whether orders are on time (and when evaluating the costs), we know the time at which the service starts so we can request the sample that belongs to the time period from the sample matrix.

Implementing time dependent service times in POA is more difficult. For time dependent travel times we use a transition calendar for which you can set which 'distribution' to use in which time period. There exists no similar participation calendar where you could set multiple time periods and for each interval you have a probability distribution. This could partially be solved by using a calendar to reduce the capacity outside the time window. In this way the service times take longer compared to service times within the time window. What we are doing here is stretching the distribution instead of using another distribution.

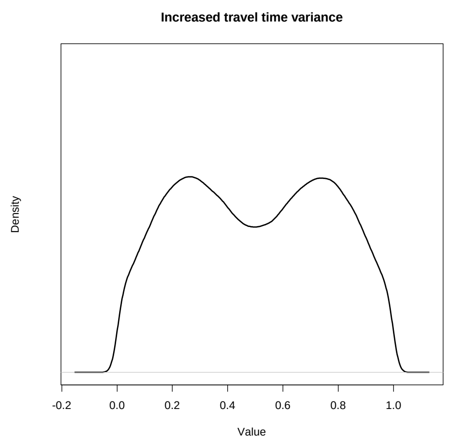
Service reliability requirement The service reliability requirement is defined as $factor \cdot (violation + offset)^{power}$ where the violation counts the number of simulations which violate the time window. This means we do not take into account the amount of lateness in every simulation world. If we have two orders and both of them have a reliability of 0, i.e. 95% violation (assuming service requirement of 95%), then we can still have very different latenesses. It can be that the first order is 30 minutes late in expected time, while the other is 10 hours late. The penalty for the travel time distribution optimizer is equal for these two orders. This is the idea of hard time windows: you are late or you are not, but if you are late we do not care how much.

Future research is to investigate if taking the amount of lateness in the simulation worlds into account gives better transportation plans. This could be proposed to transportation companies to see what they deem more important.

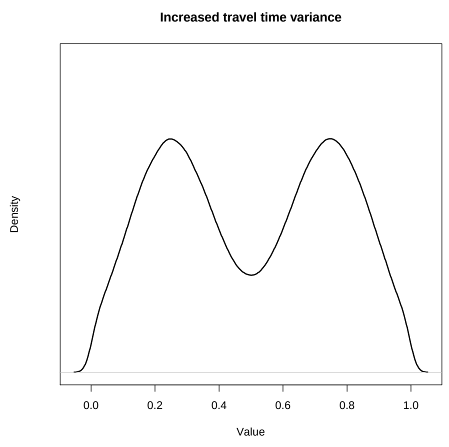
Increased travel time variance We would like to investigate how the proposed solution approaches behave under increased travel time variance. Figure 8.1 shows that we can sample from a mixture of normal distributions instead of sampling from a uniform distribution when generating simulation worlds to get travel times with more variance.

Auto-tuning service requirement weight We would like to investigate if the proposed solution approaches can be improved by auto-tuning the relative relative weight of the cost of not meeting a service reliability requirement. We expect this performs better than trying to choose a fixed value for that weight.

Improve convergence Sample Average Approximation We have seen in Section 7.3.10 that using Sample Average Approximation requires much more time than using deterministic travel times (time window slack and travel time slack). Research is needed to improve the speed of the optimization algorithm that uses Sample Average Approximation. One improvement is starting the optimization with a subset of the simulations and only if there are violations we increase the number of simulation worlds. This allows to do more iterations with a subset of simulations to get to a good result faster.



(a) Increased variance-1 for the travel times.



(b) Increased variance-2 for the travel times.

Figure 8.1: Increasing the travel time variance by sampling from a mixture of normal distributions instead of a uniform distribution.

A second improvement is related to POA. The problem is initialized in POA at the start of every iteration. This takes a significant part of the optimization duration due to the initialization of the simulation words, while the travel times in these simulation worlds do not change. Initializing this once at the start and reusing it at every iteration can improve the speed of the optimizers up to 20% we believe.

References

Bent, R. W., & Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, *52*(6), 977–987.

Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations research*, *52*(1), 35–53.

Brickell, J., Dhillon, I. S., Sra, S., & Tropp, J. A. (2008). The metric nearness problem. *SIAM Journal on Matrix Analysis and Applications*, *30*(1), 375–396.

Carrabs, F., Cordeau, J.-F., & Laporte, G. (2007). Variable neighborhood search for the pickup and delivery traveling salesman problem with lifo loading. *INFORMS Journal on Computing*, *19*(4), 618–632.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, *12*(4), 568–581.

Conijn, B. (2013). Modeling and Solving the Vehicle Routing Problem with Stochastic Travel Times and Hard Time Windows. *Eindhoven University of Technology, Department of Mathematics and Computer Science.*

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

Evers, L. (2013). Phd-project robust and agile planning. *Newsletter Expertise Centre Military Operations Research, June*.

Goel, A., & Kok, L. (2012). Truck driver scheduling in the United States. *Transportation science*, *46*(3), 317–326.

Gorissen, B. L., Yanıkoğlu, İ., & den Hertog, D. (2015). A practical guide to robust optimization. *Omega*, *53*, 124–137.

Gupta, A., & Maranas, C. D. (2003). Managing demand uncertainty in supply chain planning. *Computers & Chemical Engineering*, *27*(8), 1219–1227.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European journal of operational research*, *144*(2), 379–396.

Kok, A. L., Hans, E., Schutten, J., & Zijm, W. (2010). Vehicle routing with traffic congestion and drivers' driving and working rules.

Lee, J. H. (2014). Energy supply planning and supply chain optimization under uncertainty. *Journal of Process Control*, *24*(2), 323–331.

Lei, H., Laporte, G., & Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, *38*(12), 1775–1783.

Lekkerkerker, M. (2016). *Robust scheduling of the vehicle routing problem with time windows* (Master's thesis). Retrieved from `http://dspace.library.uu.nl/bitstream/handle/1874/340055/thesis.pdf`

Lint, J., & Zjipp, N. (2003). An improved travel time estimation algorithm using dual loop detectors. In *Trb 82nd annual meeting. washington dc.*

Maggioni, F., Potra, F. A., & Bertocchi, M. (2015). Stochastic versus Robust Optimization for a Transportation Problem. *Optimization Online*, 03–4805.

Mula, J., Poler, R., Garcia-Sabater, J., & Lario, F. C. (2006). Models for production planning under uncertainty: A review. *International journal of production economics*, *103*(1), 271–285.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, *225*(1), 1–11.

Pisinger, D., & Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics* (pp. 399–419). Springer.

Postek, K., Ben-Tal, A., Den Hertog, D., & Melenberg, B. (2015). Exact robust counterparts of ambiguous stochastic constraints under mean and dispersion information.

Prescott-Gagnon, E., Desaulniers, G., Drexl, M., & Rousseau, L.-M. (2010). European driver rules in vehicle routing with time windows. *Transportation Science*, *44*(4), 455–473.

Quintiq vehicle routing problem with time windows. (n.d.). `http://www.quintiq.com/optimization/vrptw-world-records.html`. (Accessed: 2016-06-28)

Rego, C., Gamboa, D., Glover, F., & Osterman, C. (2011). Traveling salesman problem heuristics: leading methods, implementations and latest advances. *European Journal of Operational Research*, *211*(3), 427–441.

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*(4), 455–472.

Sharifzadeh, M., Garcia, M. C., & Shah, N. (2015). Supply chain network design and operation: Systematic decision-making for centralized, distributed, and mobile biofuel production using mixed integer linear programming (MILP) under uncertainty. *Biomass and Bioenergy*, *81*, 401–414.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming* (pp. 417–431).

Solomon benchmark. (1987). `https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/`. (Accessed: 2016-06-28)

Taş, D., Dellaert, N., Van Woensel, T., & De Kok, T. (2013). Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, *40*(1), 214–224.

European Union. (2006, March 15). Regulation (EC) No. 561/2006 [REGULATION (EC) No 561/2006 on the harmonisation of certain social legislation relating to road transport and amending Council Regulations (EEC) No 3821/85 and (EC) No 2135/98 and repealing Council Regulation (EEC) No 3820/85]. *Official Journal of the European Union.*

Tomtom custom travel times api. (n.d.). `http://developer.tomtom.com/products/realtimemaps/traffic/trafficstats/TrafficStats_Custom_Travel_Times`. (Accessed: 2016-10-20)

Verweij, B., Ahmed, S., Kleywegt, A. J., Nemhauser, G., & Shapiro, A. (2003). The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, *24*(2-3), 289–333.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, *234*(3), 658–673.

Appendix A

Format communication Quintiq and TU/e

A.1 Example request

An example request file is below:

```
--- Start of file ---
Eindhoven,51.441642,5.469722
Amsterdam,52.370216,4.895168

GROUP1,Eindhoven
GROUP2,Amsterdam

GROUP1, GROUP2
--- End of file ---
```

A.2 Example response

An example response for the above request. Note that the below file has 6 lines and that indented lines are wrapped from the previous line.

```
--- Start of file ---
Origin,Destination,Day,Time,Duration,Average,0.05,0.1,0.15,0.2,0.25,0.3,
        0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95
Eindhoven,Amsterdam,WorkDays,00:00:00,07:15:00,01:27:04,01:35:07,01:35:46,
        01:36:16,01:36:43,01:37:06,01:37:29,01:37:52,01:38:15,01:38:42,
        01:39:11,01:39:42,01:40:14,01:40:55,01:41:43,01:42:42,01:44:11,
        01:46:21,01:50:11,02:00:48
Eindhoven,Amsterdam,WorkDays,07:15:00,02:15:00,01:41:45,01:36:10,01:36:58,
        01:37:37,01:38:13,01:38:49,01:39:24,01:40:00,01:40:41,01:41:24,
        01:42:12,01:43:06,01:44:13,01:45:37,01:47:25,01:49:53,01:53:37,
        01:59:54,02:13:06,03:02:27
Eindhoven,Amsterdam,WorkDays,09:30:00,07:30:00,01:29:55,01:36:28,01:37:21,
        01:38:03,01:38:39,01:39:14,01:39:49,01:40:25,01:41:08,01:41:50,
        01:42:42,01:43:40,01:44:47,01:46:08,01:47:45,01:49:58,01:52:59,
        01:56:25,02:01:34,02:14:16
Eindhoven,Amsterdam,WorkDays,17:00:00,02:00:00,01:36:00,01:36:34,01:37:29,
        01:38:14,01:38:55,01:39:33,01:40:10,01:40:50,01:41:35,01:42:28,
        01:43:25,01:44:33,01:45:46,01:47:28,01:49:34,01:52:37,01:57:04,
        02:03:43,02:15:22,02:42:06
Eindhoven,Amsterdam,WorkDays,19:00:00,04:59:00,01:24:06,01:35:55,01:36:42,
        01:37:17,01:37:49,01:38:19,01:38:48,01:39:18,01:39:48,01:40:25,
        01:41:01,01:41:38,01:42:31,01:43:24,01:44:26,01:45:53,01:47:57,
        01:50:19,01:55:06,02:06:06
--- End of file ---
```

Appendix B

Additional experimental results

The tables in this appendix give additional results from the conducted experiments. Results with a * denote the average result of 3 runs, which can cause rational numbers for routes and orders.

We give an overview of the meaning of the various columns used in the tables of this appendix:

Δ
: How much worse is the solution than the best known solution (in %).

Virtual Stochastic Costs
: The *Virtual Stochastic Costs* as defined in Section 3.4.4.

Virtual Expected Costs
: The *Virtual Expected Costs* as defined in Section 5.3.3.3.

#R
: The number of routes in the solution.

Costs
: The costs of the transportation plan excluding the penalty costs incurred by constraint violations. That is, the costs for vehicle and driver usage as explained in Sections 3.4.1 and 3.4.2.

Dist.
: Total distance driven by the vehicles in the transportation plan.

#unrel. orders
: The number of orders which do not satisfy the service reliability requirement as defined in Section 3.3.2.

Largest violat. unrel. order
: An order which satisfies the service reliability requirement has a violation. This column gives the violation of the order with the largest violation of the service reliability requirement. The violation is always between 0 (meet the service reliability requirement) and 1 (never serve the order on time). Note

that a violation of 0.5 can still mean that the order is never served on time due to a service reliability requirement of 0.5.

#unrel. routes
The number of routes which do not satisfy the driving limit reliability requirement as defined in Section 3.3.5.

Largest violat. unrel. route
A route which does not satisfy the driving limit reliability requirement has a violation. This column gives the violation of the route with the largest violation of the driving limit reliability requirement. The violation is always between 0 (meet the driving limit reliability requirement) and 1 (the driving time is never below the driving limit). Note that a violation of 0.5 can still mean that the order is never served on time due to a service reliability requirement of 0.5.

Sum violat. unrel. routes
Connected to the previous column. While the previous column gives the largest violation, this column gives the sum of the violations.

Solution approach
Which solution approach is used to obtain the transportation plan.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	12,896	11,365	30	11,365	7,320	1	0.29	2	0.09	0.18	Best known
*	190%	37,462	9,602	22.33	9,602	5,754	83	0.44	9	0.39	1.37	Expected time optimization 1h
*	3%	13,291	11,691	32	11,639	7,415	2	0.29	2	0.12	0.19	Travel time distribution optimization 1h

Table B.1: Problem instance *V-357* using expected time optimizer and travel time distribution optimizer. * denotes the average result of 3 runs and ● gives the best known result using the travel time distribution optimizer.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	12,896	11,365	30	11,365	7,320	1	0.29	2	0.09	0.18	Best known
	111%	27,245	9,889	24	9,889	5,945	65	0.29	9	0.25	0.99	30 min.
	75%	22,632	10,534	26	10,534	6,564	41	0.29	10	0.20	0.58	60 min.
	60%	20,614	10,977	29	10,977	6,923	28	0.29	10	0.14	0.49	90 min.
*	60%	20,665	12,226	33.3	12,071	7,777	27.7	0.29	7.7	0.22	0.47	120 min.
*	52%	19,561	12,548	35.3	12,443	7,960	23	0.29	5.7	0.19	0.52	150 min.
*	**50%**	19,309	13,320	42	13,320	8,598	13	0.29	7.3	0.14	0.37	180 min.
	59%	20,506	14,398	48	14,398	9,289	9	0.29	9	0.14	0.39	210 min.

Table B.2: Problem instance *V-357* using time window slack. The amount of slack is in minutes. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for the time window slack.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	12,896	11,365	30	11,365	7,320	1	0.29	2	0.09	0.18	Best known
	112%	27,290	10,220	24	10,076	6,026	60	0.29	9	0.19	0.86	percentage = 10%
	43%	18,407	11,168	26	10,953	6,541	25	0.16	5	0.38	0.58	percentage = 30%
*	35%	17,398	11,624	28.7	11,624	6,752	17	0.07	5.7	0.18	0.51	percentage = 50%
*	**26%**	16,312	12,292	30.7	12,292	7,008	4.7	0.04	6.3	0.15	0.42	percentage = 70%
*	28%	16,519	13,012	32	12,959	7,118	3.3	0.03	5.7	0.18	0.42	percentage = 90%
	31%	16,929	13,700	34	13,700	7,464	1	0.02	6	0.14	0.37	percentage = 110%
	37%	17,667	14,780	37	14,780	7,854	0	0	6	0.11	0.27	percentage = 130%
	69%	21,777	10,519	26	10,519	6,254	39	0.19	9	0.09	0.38	slack = Std * 0.5
*	**18%**	15,245	11,280	28	11,280	6,544	9	0.05	4.7	0.16	0.41	slack = Std * 1.0
*	17%	15,135	12,133	31	12,085	6,953	3.7	0.03	4.7	0.14	0.35	slack = Std * 1.5
*	29%	16,646	13,604	34	13,604	7,434	0	0	6	0.13	0.35	slack = Std * 2.5
*	44%	18,625	15,410	40	15,410	8,221	0	0	6.7	0.11	0.31	slack = Std * 3.5
	52%	19,665	17,239	41	17,239	8,660	0	0	5	0.10	0.24	slack = Std * 4.5
	79%	23,129	19,956	48	19,956	9,990	0	0	7	0.05	0.22	slack = Std * 5.5
	102%	26,106	10,257	24	10,257	6,161	59	0.24	10	0.15	0.53	slack = MAD * 0.5
*	19%	15,329	11,204	27.7	11,204	6,512	12	0.07	4	0.16	0.40	slack = MAD * 1.5
*	**17%**	15,041	12,296	31	12,296	7,106	2	0.04	4.7	0.11	0.31	slack = MAD * 2.5
*	21%	15,606	13,014	32.7	13,014	7,277	0.7	0.01	5	0.11	0.28	slack = MAD * 3.5
	29%	16,615	14,466	36	14,466	7,841	0	0	4	0.14	0.29	slack = MAD * 4.5
	38%	17,832	15,339	40	15,339	8,373	0	0	5	0.11	0.27	slack = MAD * 5.5

Table B.3: Problem instance *V-357* using travel time slack. The first set uses travel time slack with a fixed percentage. The travel time slack in the second set is based on the standard deviation and for the third set the slack is based on the MAD. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for each of the three types of slack.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	17,211	14,371	37	14,371	9,093	3	0.29	4	0.07	0.26	Best known
*	230%	56,822	11,175	25.67	11,175	7,022	134	0.55	10.67	0.32	1.93	Expected time optimization 1h
*	4%	17,977	15,521	41.33	15,521	9,881	4	0.29	2.67	0.09	0.21	Travel time distribution optimization 1h

Table B.4: Problem instance *V-357 tight-50* using expected time optimizer and travel time distribution optimizer. * denotes the average result of 3 runs and ● gives the best known result using the travel time distribution optimizer.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	17,211	14,371	37	14,371	9,093	3	0.29	4	0.07	0.26	Best known
	152%	43,313	12,037	27	11,845	7,503	118	0.29	12	0.37	1.63	30 min.
	99%	34,278	12,979	30.0	12,727	8,171	76	0.29	13	0.46	1.16	60 min.
*	91%	32,957	14,639	35.0	14,571	9,456	63.7	0.29	14.3	0.26	0.93	90 min.
*	**63%**	28,117	17,616	45	17,556	10,707	35.3	0.29	9	0.19	0.56	120 min.
*	68%	28,832	20,996	56.7	20,996	12,091	20	0.29	9	0.14	0.49	150 min.
	86%	31,973	25,229	74	25,229	13,981	14	0.29	9	0.08	0.37	180 min.
	101%	34,668	27,900	89	27,900	15,148	11	0.29	10	0.12	0.39	210 min.

Table B.5: Problem instance *V-357 tight-50* using time window slack. The amount of slack is in minutes. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for the time window slack.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	17,211	14,371	37	14,371	9,093	3	0.29	4	0.07	0.26	Best known
	136%	40,588	11,911	27	11,726	7,112	110	0.33	11	0.25	1.05	percentage = 10%
	76%	30,322	12,951	30	12,754	7,472	74	0.18	8	0.29	0.73	percentage = 30%
	45%	25,030	14,068	32.0	14,068	7,939	39	0.12	10	0.18	0.49	percentage = 50%
*	31%	22,536	15,810	35.3	15,810	8,488	18.7	0.07	8.3	0.11	0.31	percentage = 70%
*	**27%**	21,866	16,692	36.7	16,632	8,710	8.3	0.06	7	0.23	0.54	percentage = 90%
*	30%	22,428	18,036	39.7	18,036	8,953	5	0.03	7.3	0.11	0.38	percentage = 110%
*	31%	22,529	19,025	41	19,025	9,172	1	0.01	6.67	0.12	0.38	percentage = 130%
	87%	32,172	13,016	30	12,831	7,749	83	0.22	8	0.29	0.74	slack = Std * 0.5
*	36%	23,350	13,874	31.3	13,754	7,800	32	0.11	8.3	0.23	0.58	slack = Std * 1.0
*	**17%**	20,110	15,423	35.3	15,376	8,583	10.7	0.07	6.3	0.13	0.35	slack = Std * 1.5
*	25%	21,543	17,977	40.7	17,977	9,056	0	0	7.3	0.11	0.35	slack = Std * 2.5
*	37%	23,498	20,250	44.3	20,250	9,559	0	0	7.0	0.11	0.25	slack = Std * 3.5
	49%	25,656	22,617	47	22,617	10,148	0	0	6	0.10	0.36	slack = Std * 4.5
	71%	29,444	26,253	52	26,253	11,261	0	0	7	0.05	0.23	slack = Std * 5.5
	126%	38,957	12,191	28	12,191	7,462	101	0.3	13	0.13	0.84	slack = MAD * 0.5
	41%	24,262	13,744	32.0	13,557	7,638	35	0.11	10	0.31	0.62	slack = MAD * 1.5
*	**19%**	20,504	15,727	36.3	15,588	8,562	8	0.05	6.7	0.29	0.50	slack = MAD * 2.5
*	23%	21,179	17,141	38.0	17,071	8,873	1.0	0.02	7.7	0.19	0.45	slack = MAD * 3.5
*	27%	21,866	18,984	42.3	18,930	9,590	0	0	5.7	0.15	0.35	slack = MAD * 4.5
	40%	24,149	20,196	44	20,196	9,649	0	0	8	0.11	0.42	slack = MAD * 5.5

Table B.6: Problem instance *V-357 tight-50* using travel time slack. The first set uses travel time slack with a fixed percentage. The travel time slack in the second set is based on the standard deviation and for the third set the slack is based on the MAD. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for each of the three types of slack.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	16,575	14,167	36	14,167	8,961	7	0.29	2	0.07	0.11	Best known
*	183%	46,896	11,167	25	11,103	6,917	106	0.52	10.67	0.35	1.59	Expected time optimization 1h
*	8%	17,893	15,342	41.0	15,342	9,938	7	0.29	2	0.13	0.21	Travel time distribution optimization 1h

Table B.7: Problem instance *V-357 service* using expected time optimizer and travel time distribution optimizer. * denotes the average result of 3 runs and ● gives the best known result using the travel time distribution optimizer.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	16,575	14,167	36	14,167	8,961	7	0.29	2	0.07	0.11	Best known
	118%	36,060	12,219	28	11,996	7,544	84	0.29	13	0.36	1.32	30 min.
	85%	30,730	12,808	31	12,808	8,344	57	0.29	13	0.25	1.20	60 min.
*	73%	28,734	14,576	34.3	14,576	9,343	38.7	0.29	13.7	0.17	0.94	90 min.
*	**68%**	27,798	17,393	45	17,343	10,502	28.7	0.29	11	0.19	0.65	120 min.
*	77%	29,339	20,673	51	20,673	11,732	19.3	0.29	10.7	0.17	0.59	150 min.
	96%	32,432	25,231	74	25,231	13,792	17	0.29	9	0.08	0.38	180 min.
	107%	34,321	27,795	84	27,795	14,782	15	0.29	8	0.12	0.36	210 min.

Table B.8: Problem instance *V-357 service* using time window slack. The amount of slack is in minutes. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for the time window slack.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	16,575	14,167	36	14,167	8,961	7	0.29	2	0.07	0.11	Best known
	107%	34,279	15,234	28.0	11,864	7,286	76.0	0.38	10.0	0.25	1.00	percentage = 10%
	60%	26,580	13,490	30.0	13,263	7,806	49	0.24	7	0.40	0.75	percentage = 30%
*	45%	23,957	14,322	32.3	14,276	7,947	35.7	0.14	7.3	0.17	0.56	percentage = 50%
*	**35%**	22,454	15,675	35.3	15,675	8,492	17.7	0.1	8.3	0.12	0.41	percentage = 70%
*	42%	23,456	16,475	37	16,475	8,611	14.3	0.07	9.3	0.22	0.56	percentage = 90%
	38%	22,810	18,059	40	18,059	8,991	9	0.08	7	0.12	0.35	percentage = 110%
	39%	23,100	19,015	41	19,015	9,216	2	0.03	8	0.11	0.34	percentage = 130%
	78%	29,518	12,754	29.0	12,569	7,476	64	0.22	10	0.29	0.68	slack = Std * 0.5
*	36%	22,466	13,757	30.7	13,637	7,736	32	0.1	7.3	0.23	0.49	slack = Std * 1.0
*	**21%**	19,999	15,329	35	15,237	8,490	11.3	0.1	5.7	0.15	0.41	slack = Std * 1.5
*	33%	22,083	17,479	38.7	17,432	8,805	4.3	0.02	8.3	0.14	0.39	slack = Std * 2.5
*	44%	23,904	20,427	45	20,427	9,723	0.3	0	7.3	0.13	0.28	slack = Std * 3.5
	60%	26,543	22,786	48	22,786	10,266	0	0	8	0.10	0.32	slack = Std * 4.5
	80%	29,861	26,237	52	26,237	11,346	0	0	8	0.05	0.25	slack = Std * 5.5
	78%	29,439	12,099	27	12,099	7,238	61	0.22	11	0.13	0.75	slack = MAD * 0.5
*	36%	22,564	13,789	31	13,603	7,725	31.3	0.12	7.3	0.30	0.53	slack = MAD * 1.5
*	**23%**	20,449	15,402	35.3	15,341	8,396	10.3	0.08	6.7	0.20	0.47	slack = MAD * 2.5
*	30%	21,524	16,880	38.3	16,880	8,752	4.3	0.03	8	0.16	0.44	slack = MAD * 3.5
	32%	21,856	19,211	42	19,211	9,468	1	0.01	5	0.11	0.28	slack = MAD * 4.5
	50%	24,844	20,371	45	20,371	9,722	4	0.01	8	0.11	0.41	slack = MAD * 5.5

Table B.9: Problem instance *V-357 service* using travel time slack. The first set uses travel time slack with a fixed percentage. The travel time slack in the second set is based on the standard deviation and for the third set the slack is based on the MAD. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for each of the three types of slack.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	21,674	14,466	39	14,466	9,283	3	0.29	13	0.16	0.70	Best known
*	175%	59,704	11,189	25	11,189	6,951	140	0.64	13.3	0.34	2.14	Expected time optimization 1h
*	3%	22,231	15,406	41.3	14,887	9,289	4.7	0.32	12.0	0.13	0.74	Travel time distribution optimization 1h

Table B.10: Problem instance *V-357 driving* using expected time optimizer and travel time distribution optimizer. * denotes the average result of 3 runs and ● gives the best known result using the travel time distribution optimizer.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	21,674	14,466	39	14,466	9,283	3	0.29	13	0.16	0.70	Best known
	109%	45,326	12,088	27	11,819	7,395	122	0.29	13	0.57	2.19	30 min.
	75%	37,970	12,866	30	12,593	8,010	81	0.29	15	0.62	1.91	60 min.
*	65%	35,854	14,723	34.7	14,668	9,566	59.7	0.29	20	0.24	1.56	90 min.
*	**60%**	34,610	17,555	49	17,456	10,637	36.3	0.29	21.3	0.25	1.36	120 min.
*	69%	36,666	21,105	55.3	21,105	12,129	20.3	0.29	25	0.18	1.15	150 min.
	85%	40,102	25,190	76	25,190	13,736	12	0.29	27	0.12	1.07	180 min.
	102%	43,770	27,932	87	27,932	15,081	13	0.29	29	0.16	1.06	210 min.

Table B.11: Problem instance *V-357 driving* using time window slack. The amount of slack is in minutes. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for the time window slack.

	Δ	Virtual Stochastic Costs	Virtual Expected Costs	#R	Costs	Dist.	#unrel. orders	Largest violat. unrel. order	#unrel. routes	Largest violat. unrel. route	Sum violat. unrel. routes	Solution approach
●	0%	21,674	14,466	39	14,466	9,283	3	0.29	13	0.16	0.70	Best known
	114%	46,487	11,878	26	11,688	7,067	118	0.35	15.0	0.35	1.78	percentage = 10%
	40%	30,376	13,399	30	13,173	7,718	54	0.19	13	0.44	1.15	percentage = 30%
*	32%	28,601	14,202	31.3	14,098	7,849	36.3	0.11	16.7	0.21	1.11	percentage = 50%
*	**23%**	26,643	15,716	36	15,716	8,589	15.3	0.08	17.7	0.15	0.84	percentage = 70%
*	23%	26,686	16,630	37.3	16,630	8,717	6	0.05	18.3	0.25	0.96	percentage = 90%
	35%	29,308	18,352	42	18,352	9,235	4	0.04	22	0.15	0.87	percentage = 110%
	34%	28,958	18,734	40	18,734	9,089	2	0.01	20	0.24	1.02	percentage = 130%
	64%	35,648	12,749	29.0	12,564	7,483	78	0.3	15	0.33	1.18	slack = Std * 0.5
*	27%	27,560	13,890	32	13,769	7,816	37	0.09	15.67	0.29	0.93	slack = Std * 1.0
*	**17%**	25,417	15,460	36.3	15,460	8,670	9.7	0.08	17.7	0.17	0.77	slack = Std * 1.5
*	17%	25,374	17,515	39.3	17,515	8,869	0.0	0	16	0.14	0.81	slack = Std * 2.5
*	34%	29,148	20,537	45.7	20,537	9,690	0.0	0	18.3	0.15	0.72	slack = Std * 3.5
	47%	31,766	23,085	49	23,085	10,411	0	0	18	0.14	0.82	slack = Std * 4.5
	70%	36,780	26,183	52	26,183	11,040	0	0	23	0.09	0.80	slack = Std * 5.5
	86%	40,386	12,249	28	12,249	7,549	106	0.29	14	0.22	1.40	slack = MAD * 0.5
*	24%	26,913	13,697	30.7	13,511	7,694	35	0.11	15	0.33	0.91	slack = MAD * 1.5
*	**15%**	24,942	15,698	35.7	15,633	8,538	7.3	0.05	16.3	0.24	0.89	slack = MAD * 2.5
*	16%	25,176	17,062	38.3	17,062	8,801	0.3	0.01	16	0.19	0.90	slack = MAD * 3.5
	27%	27,526	18,919	42	18,919	9,478	0	0	18	0.15	0.77	slack = MAD * 4.5
	32%	28,709	20,336	45	20,336	9,917	0	0	17	0.15	0.87	slack = MAD * 5.5

Table B.12: Problem instance *V-357 driving* using travel time slack. The first set uses travel time slack with a fixed percentage. The travel time slack in the second set is based on the standard deviation and for the third set the slack is based on the MAD. * denotes the average result of 3 runs, ● gives the best known result and the bold result shows the best score for each of the three types of slack.