University of Twente

Industrial Engineering and Management

Master Thesis

Improving the Planning Process of a Complex Pickup & Delivery Problem by Applying a Constructive Heuristic and an Adaptive Large Neighbourhood Search

Author:
B. Reitsma

First Supervisor:
dr. ir. W.J.A. Van Heeswijk
Second Supervisor:
dr. ir. A.G. Leeftink
Company Supervisor:
ir. J. Van Heuveln

October, 2022

UNIVERSITY OF TWENTE.

Management Summary

This research is performed for Company X, a transport company of goods for the medical sector with a fleet of ± 45 trucks and ± 5.000 daily orders operating mainly in Belgium. Company X is a client of CAPE Groep, which develops a new planning environment for them. Multiple problems within the planning process are caused by a lack of digital support for the planning personnel. These problems result in unsatisfactory transport schedules in terms of routing costs and time and effort spend to create these schedules. Therefore, we formulated the following main research question:

"How can a decision support system for transport route scheduling help Company X's planning personnel to create schedules faster and more efficient with lower routing costs?"

Problem Context
In the current situation, the planning process consists of six steps. From the six planning steps within their planning process, improvements within the distribution planning are most beneficial. Therefore, we reduced our scope to the distribution planning. In this planning step, planners create routes for vehicles based on the orders of customers. The three reasons to focus on this planning are: 1) It is the most experience-dependent planning step because of lack of (digital) support for routing decision and planning constraints; 2) This step is the most labour-intensive, and therefore it requires the most labour hours to execute; 3) Besides the distribution planning, the same planners also do the recovery planning afterwards, with reduced labour productivity as a result. In the recovery planning, planners change the distribution planning when faced with unforeseen events at night. Currently, the required labour hours are already mitigated by using predefined routes of order locations. Because standard routes with corresponding standard orders reduce the number of planning decisions. However, the routing cost-effectiveness of the standard orders is unclear because of volume fluctuation and customer changes.

Solution Design
The problem context of Company X translates into a newly defined complex theoretical routing problem, the Multi-Trip Multi-Compartment Pickup and Delivery Problem Time Windows (MTMCPDPTW). The mathematical problem formulation of this problem cannot be solved exactly in reasonable time with the context and instance of the routing problem of Company X. Therefore, we designed two solution approaches to create a distribution planning: a fast running constructive algorithm using the concepts of a savings algorithm and an Adaptive Large Neighbourhood Search (ALNS) improvement algorithm tailored to Company X. The constructive algorithm uses standard orders as a base schedule and assigns the remaining capacity by the most savings with insertion. The ALNS contains four destroy and repair heuristics to find an improved solution in a given time.

Experiment Results
We conducted experiments with a dataset containing the orders of a representative average day for Company X. From experimenting with different ALNS parameter input settings, we found, besides good performing input settings, that randomness plays a role in the performance of the ALNS. To mitigate the effect of randomness, we performed an additional experiment with different randomness inputs on two good performing settings. The results were statistically insignificant with overlapping confidence intervals ($\alpha = 5\%$) and a two-sample t-test on the sample mean with a p-value of 0.22. Nevertheless, we recommend the setting with the lowest mean and standard deviation, which has [€37.883, €38.402] as the confidence interval on the mean routing costs.

We compare the result of the manual planned schedule with the constructive algorithm and the improvement algorithm with the recommended setting. For all schedules, we determine the total cost of a schedule balancing the driver costs, kilometre costs and costs of not serving pallets. We conclude for Company X:

- A cost reduction up to 26% (on average 24%) is achievable by using the designed solutions.
- Only 14% of the orders in standard routes are cost-effective.

The manual solution has total costs of ≈€50.000, whereas the constructive and improvement design have costs of ≈€42.500 (-15%) and ≈€38.000 (-24%) respectively. The improvement algorithm shows that a solution with low costs only uses 17 out of the 125 predefined standard orders in the route, and

therefore we recommend to further investigate the effectiveness of the orders in standard routes within the distribution planning.

Implementation
To enable usage in practice, we built a data architecture for the solution designs, allowing implementation in a dedicated decision support system. Further research should focus on implementation in the new planning environment of Company X. For using the solution designs in practice, we described two decision structures for two different use cases. The first decision structure is for building a solution from scratch, an iterative evaluation procedure using both the constructive algorithm and the ALNS improvement algorithm. The structure for the second use case is for repairing a schedule with an unavailable vehicle. This decision structure is based on using the improvement algorithm with different settings. The effectiveness of this last decision structure has yet to be tested. We hypothesize that it procedures routes of at least equal quality in less time.

Practical contribution
The main practical contribution of this thesis is that Company X can reduce total costs of their routes with up to 26% by applying the designed solutions in their planning process. To achieve cost reductions together with creating faster and more efficient schedules in a practical environment, the solutions are worked out in decision structures. The second contribution is an analysis of standard routes, concluding that most orders of these standard routes are suboptimal and thus should be evaluated by Company X. The last practical contribution is the elaborate description of the data architecture together with algorithmic descriptions of the constructive algorithm and improvement algorithm with all used heuristics. This enables researchers and developers of CAPE Groep to solve other routing problems with the same solution design. Application in other sectors than the transport of medical goods, in which Company X operates, is possible.

Scientific contribution
This thesis contributes to the literature in various ways. The first contribution is the definition and theoretical formulation of the MTMCPDPTW. Secondly, we have shown that this formulation of the problem is not solvable exactly in reasonable time for problems with more than six customers. The last contribution is that solving the MTMCPDPTW with ALNS in the context of Company X requires $\approx$100 times more computational power than other ALNS algorithms for PDPs in literature. For further research, we therefore recommend studying parallel solving methods or changes to the algorithm reducing computation time. The changes we suggest are: parallel solving methods, smaller repair neighbourhoods or estimated insertion costs.

Contents

Abbreviations

ACO Ant Colony Optimization algorithm.

ADR Accord européen relatif au transport international des marchandises Dangereuses par Route.

ALNS Adaptive Large Neighbourhood Search.

CCP Chance Constrained Program.

Company X transport company of medical goods located in the north of Belgium.

DARP Dial A Ride Problem.

DVRP Dynamic Vehicle Routing Problem.

ERP Enterprise Resource Planning.

GA Genetic Algorithm.

ILP Integer Linear Program.

LDP Logistic Dynamic Planning.

MCVRP Multi-Compartment Vehicle Routing Problem.

MDVRP Multi-Depot Vehicle Routing Problem.

MTMCPDPTW Multi-Trip Multi-Compartment Pickup and Delivery Problem Time Windows.

MTVRP Multi-Trip Vehicle Routing Problem.

NAT Network Analysis Tool.

OOMHPA Operational Order Management Health Process Application.

OVRP Open Vehicle Routing Problem.

PDP Pickup and Delivery Problem.

PSO Particle Swarm Optimization.

SA Simulated Annealing.

SBRP School Bus Routing Problem.

SPR Stochastic Program with Recourse.

subco subcontractor planning.

SVRP Stochastic Vehicle Routing Problem.

TDVRP Time Dependent Vehicle Routing Problem.

TS Tabu Search.

TSP Traveling Salesman Problem.

TUP Transport Uitvoerende Partij (transport executing company).

UI User Interface.

VNS Variable Neighbourhood Search.

VRP Vehicle Routing Problem.

VRPB Vehicle Routing Problem Backhauls.

VRPTW Vehicle Routing Problem Time Windows.

List of Figures

List of Tables

1 Introduction

Section 1.1 of this research introduces CAPE Groep and the client for which the research will be conducted. Section 1.2 will explain the identification of the problem at the client. Subsequently, Section 1.3 discusses the relevancy of this research to theory. The research goal and the questions are given in Section 1.4. Lastly, the outline of the remainder of the thesis is discussed in Section 1.5.

1.1 Background

CAPE Groep is a consultancy firm in the construction, agriculture, and logistics sector. CAPE Groep is specialised in creating and maintaining low-coding applications for its various clients. From this point onwards, CAPE Groep is referred to as Cape. The advantages of their low-code applications are rapid development and understandability for the end-user of clients. The end-user could be, for example, order planners. Cape uses the Mendix platform for the development of low-code applications. Mendix provides the possibility of developing together with developers from different organisations. Moreover, end-users can also participate in development due to the low-code developing environment.

The client for which this study is conducted is a transport company of medical goods located in the north of Belgium. From now on, we refer to this company as Company X. Company X transports medical supplies in Western Europe. Medical supplies are, in this context, medicines or vaccines or supportive materials for medical treatment such as face masks or surgical gowns. Medical supplies may have special restrictions on for example, the temperature of transport. The supplies have to be within certain temperature ranges during the whole transport process. The other restriction, Company X has to deal with is the ADR, a European agreement for transporting hazardous substances by road. These regulations are in place to ensure road safety by restricting the transport of hazardous substances on the road. In practice, this means that orders with hazardous substance receive a number of ADR points. As a result, if the sum of ADR points for a vehicle exceeds a limit, a certified driver has to drive this vehicle, which are limited available. All the factors mentioned, together with the demand for $\pm$ 5,000 orders a day, executed with $\pm$ 40 vehicles, make Company X a company with a complex transportation portfolio.

In this research, we focus on the scheduling activities of Company X (see Figure 1). These activities are either automated activities, or manual activities performed by personnel in the suitable application of Company X. An example of an automated process is determining which transport order will be done by which subcontractor. This is done automatically if an order has a destination in the assigned region of the subcontractor. Section 2.3 elaborates on this concept. An example of a manual operation is the scheduling of direct orders. Direct orders are special orders that require pickup and delivery on the same route. The decision of the planner is experienced based, because the application do not provide information about the context, in this case the vehicle route. All the decisions within the different planning steps are based on the experience of the planner of this particular step.

Company X is currently migrating to a new digital planning environment called Operational Order Management Health Process Application (OOMHPA). Currently, Company X uses different planning applications which all are responsible for a different part of the planning. The goal of implementing OOMHPA is to have an integrated system that processes all the steps from planning an incoming order to collecting the goods for distribution. One of the main requirements of OOMHPA is that it has the same functionality as the applications used in the current situation. The OOMHPA will increase the possibilities to automate or increase support for manual decision within and between the different planning steps at Company X. The OOMHPA will have more functions within Company X than just creating schedules. However, in this research, we focus on activities directly related to the transport scheduling process. In the next paragraph, we discuss scheduling activities in order to identify the problem concisely (see Figure 1). Section 2.1 elaborates more deeply these activities.

The process of creating schedules start with the creation of the resource planning. This is done every working day to determine the available resources (drivers, vehicles trailers) for the next day. The resource planning is based on a standard schedule, with orders of often reoccurring customers. The resource planning determines the start time for each route. Placeholders with a fixed volume are used when volume of an order is missing. This is to incorporate the fact that this location is likely to be visited in the schedule.

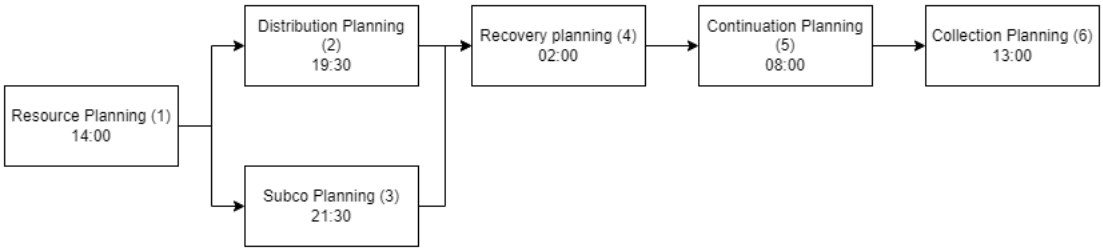


Figure 1: The start-times and sequence in which planning steps are executed at Company X

The second step is the evening schedule step. In the time between the first and second steps, all the volumes of the inserted placeholders are known. These are incorporated in the schedule and conflict orders are shuffled between routes to fix constraint violations. This step creates the distribution planning.

Simultaneously executed with Step 2 is Step 3, the subcontractor planning (subco). This is the planning containing routes which will be outsourced to subcontractors. Orders are assigned to subcontractors by fixed business rules. Per subcontractor, the orders will be divided in clusters. The route which drivers execute with the orders of the cluster is determined automatically by Logistic Dynamic Planning (LDP). The planning personnel is mainly occupied with arranging the resources from subcontractors, which is a labour-intensive process.

The fourth step in the process is the recovering scheduling step. In this step, late emerging orders are incorporated as well as changes due to calamities (illness, vehicle defects, etc.). This step is completely done manually in the current application by the scheduling personnel.

The last step takes place during the execution of the schedule in the morning, that is, the continuation planning. This process schedules additional routes for undistributed orders, collection of supplies and emergency orders. Subsequently, the standard schedule (used in the first step) for the next day is updated based on new information on the supplies received from the collection.

1.2 Problem Identification

As mentioned in the previous section, at Company X different scheduling activities are performed in the different scheduling steps. In this research, we will focus on the possible problem within the scheduling activities. In this section, we first introduce the methodology used as a guide in this process. Afterwards, we apply this to the problem context of Company X.

1.2.1 Methodology

The methodology used to guide the problem identification will be the Managerial Problem-Solving method (MPSM). This method is developed by Heerkens and van Winden (2017). The first two steps of the MPSM are about the content presented in Section 1.2 and Section 1.4. The start of the method is defining problems and clustering them in order to find the problem to solve. (Heerkens and van Winden, 2017, p. 41).

1.2.2 Defined Problems

From conversations with the scheduling personnel and the management of Company X, multiple problems were derived. These problems will be explained and afterwards presented in the problem cluster. First, we discuss all defined problems in detail, starting with the central problem. Subsequently, we discuss five numbered defined problems influencing the central problem.

The central problem can be defined as a problem Company X wants to be tackled. At Company X they are not satisfied with the schedules that are currently produced. They hope to achieve improvements in terms of routing costs, decreased scheduling time with increased efficiency. However, for them, it is difficult to define how this can be achieved. Their previously scheduling improvement project for complete automated scheduling failed because of unsatisfactory results. Therefore, it is not clear on how Company X can improve their schedules. This is the starting point of the problem identification and therefore, the

causes of this problem have to be identified to be able to tackle this problem. We define this problem as unsatisfactory transport schedules.

The first problem we present is the problem of the workload of the scheduling activities for planning personnel. As stated in Section 1.1, most activities are performed manually with the help of applications. The planning steps taking place at night (Step 2,3 and 4) are particularly burdensome for the planning personnel. For example, the recovery planning steps comes with the stress of fixing operational problems (arranging drivers, repairs to vehicles, etc.). Consequently, planning personnel cannot be scheduled efficiently at night shifts for more than two weeks in a row afterwards, another planner has to take over because of the large workload of this scheduling activity.

The second problem is related to a lack of insights into the performance of the schedule. The main goal of the schedule is to schedule as many orders as possible given the resources available. However, the effects on time spend or kilometres are driven is unknown. When changes are necessary, the effects of the changes are not evaluated. For example, allocating an order from one to another route does not indicate the change in kilometres or time for both of the routes. This type of performance insight can help to assess changes in routes. It can also help in evaluating the effects if routes are cancelled and the orders have to be rearranged to other routes.

The third problem related to the second problem is missing digital planning documentation. This is required to evaluate/compare schedules over a longer time period with more aggregated performance measures. Currently, comparing schedules can only be done on paper. Which is an impossible, or extremely time-consuming activity. Therefore, this does not happen at Company X resulting also in missing evaluation on the transport schedules.

The fourth identified problem is that scheduling activities are too experience-dependent. The definition of this problem can be best explained by some examples. An example of this experience is the incorporation of traffic into routes. Planning personnel knows, for example, that avoiding the Antwerp bypass is crucial between 16:00h and 18:00h. Another example is the knowledge of delivery destinations. Some destinations are not suitable for large vehicle types, resulting in either switching an order to another route or moving orders to other routes to be able to change the vehicle type of this route. All these types of decision are made completely based on the experience and knowledge of the planners. This type of constraints/decisions are not documented. The long-term effect of this is that training the personnel is a long process. Another effect is that schedules become worse if an inexperienced planner does the planning for that day. This makes scheduling performance highly dependent on the experience of planners.

The last problem is the inflexible outsourcing structure. Currently, outsourcing is done by creating the subco (see Section 1.1). All orders are automatically subcontracted with business logic. As stated previously, this subcontracting logic is inflexible, resulting in missed profit of outsourced orders, which could be served cheaper by Company X themselves.

1.2.3 Problem Cluster

The problems presented in the previous section are not separate problems. Some problems are related to each other or have the same consequences. Therefore, we define a problem cluster with the relations between the problems, see Figure 2. All problems are numbered for referencing purposes. Intermediate problems are problems that have another cause. For problems without an underlying problem, there are two categories. The first one being possible core problems to investigate. The second category are problems which cannot be influenced, or be solved by research. These problems are defined in the problem cluster as unworkable problems. Since there is only one possible core problem, this is automatically the core problem (Heerkens and van Winden, 2017, p. 41). In the remainder of this section, we will explain the unmentioned relations and problems in the cluster. Afterwards, we elaborate on the core problem.

One of the relations which is not mentioned before is the relation between the long training requirement (Problem 4) and the lack of personnel (Problem 9). This relation has two components. The first is that the current scheduling personnel must spend extensive time training new personnel over a longer period. This is necessary to be able to perform all experience dependent planning activities on an acceptable level. The other relation is the fact that due to the long training, not everyone is suited for the job. This limits the possible candidates and, therefore, limits the possibility of training new planners, thus causing

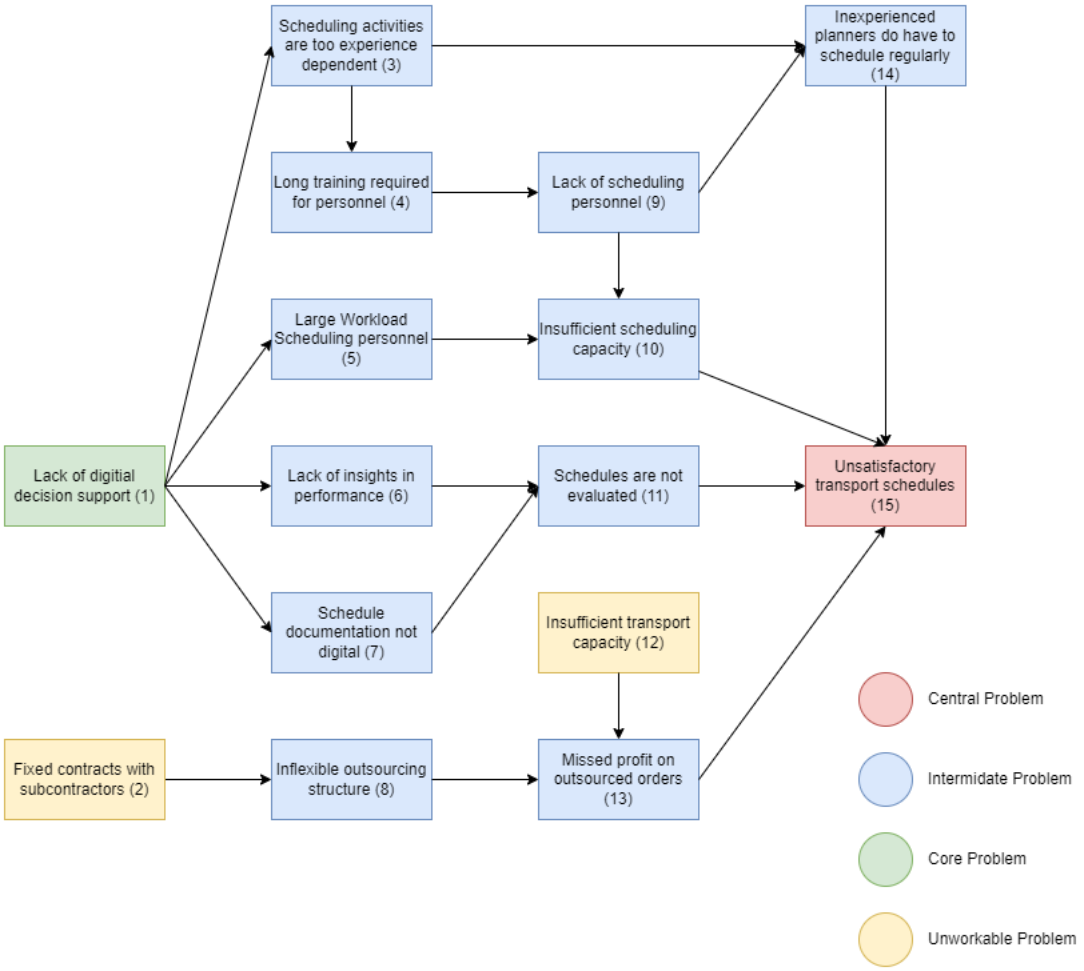


Figure 2: Problem Cluster

a lack of scheduling personnel.

Other unmentioned relations are with unworkable problems (2, 12) and their consequences (8, 13). The main cause of the inflexible outsourcing structure is the fact that the region in which subcontractors get orders are fixed, and therefore orders cannot be switched between subcontractors if that would be more sufficient. One way to mitigate this, is to perform these profitable orders for outsourcing self. However, this is not possible due to lack of capacity to do these orders themselves (problem 8). Company X is not willing to invest in this capacity and, therefore, this is an unworkable problem. More details on subcontractors in general is given in Section 2.2.

1.2.4 Core Problem

Figure 2 shows that there are 4 problems (3, 5, 6, 7) that have the same cause. The other problems (4, 8-11, 13, 14) all have different causes. The only problem with an underlying cause that is workable is problem 1. A solution to this problem can solve (partly) the problems 3, 5, 6, 7. This solution can also focus more on one of the problems to solve this completely. However, without research, we do not know which problem to focus on. Therefore, our focus stays for now on all these problems that can be solved by introducing a form of digital decision support to the routing scheduling process. In the remainder of this section, we will discuss the reasons why and how decision support can solve the core problem of the cluster.

To understand how and why a decision support can solve the problems of Company X, it is important to explain what our definition of decision support is in this context. Within the context of decision support is important to distinguish two similar terms. Decision support and decision support system/application/tool. A decision support system is a digital application which supports the user of the system in

taking decision. In the case of the problem context is that an application, which visualises routes for the fleets and information on the performance of the scheduled routes. These routes are generated by the application and can be adjusted by the end-user. This end-user also separates the decision support from the decision support system. The broader term decision support also includes the interaction and usage pattern of the user with the system.

This type of decision support is able to tackle the Problem 3 by, for example, incorporating the scheduling decision into constraints for generating the routes or providing additional information fields of, for example orders, delivery addresses when required. This reduces the dependence on experience, as information is also available to less experienced personnel or automatically dealt with. Related is the information storage Problem 7. With a decision support system, schedules can be more easily stored digitally and therefore historical schedules are accessible for evaluation at later times. Evaluation can also happen during the scheduling process with a decision support system, tackling Problem 6. The system can evaluate the effect of small changes on the whole schedule. Consequently, information on the entire schedule has to be available, hence performance indicators have to be available. These will help to gain insight into the total performance of the schedule. The size of the fifth problem will be reduced with decision support. Because of the help with crucial decisions in the schedules, the time to make decision will be reduced. Therefore, the workload for scheduling personnel will decrease.

1.3 Motivation

Route scheduling is a widely researched topic in the literature, with at least hundreds of papers published in recent years (Braekers et al., 2016). These studies mainly focus on finding new ways to create better solutions to problems with even lower costs. However, these findings are not always applied directly in practice. The literature provides some examples of explaining the difficulties in trying to bridge the gap between theoretical findings and practical use (Canen and Scott, 1995) (Yang et al., 2017). The complex scheduling process of Company X is exemplary for the difficulty to apply the numerous possibilities to improve parts of the scheduling process.

The complex contexts of problems is not often researched in literature. Most attention of researchers is going to developing improvements on existing problems, compared to applying the available techniques on more complex contexts (Braekers et al., 2016). This brings us to the scientific relevance of the thesis. An implantation of existing solving method into application to solve a complex scheduling problem. The scientific knowledge of routing in theory can be combined into the planning process of Company X to improve their transport schedules.

The practical contribution of this thesis is to introduce Company X and its planning personnel with the possibilities of the automation in scheduling by decision support. With the results and implementation of this research, Company X may reduce routing costs and spend less time in their scheduling process and do their scheduling more efficient.

1.4 Research Design

In this section, we will translate the problem described previously into a research design, starting with defining the objective of this research and continuing with the questions that need to be answered to achieve this goal. Finally, the outline of the remainder of the thesis will be discussed.

1.4.1 Research Goal

Previous sections state the potential benefits of introducing decision support and the problems it may solve for Company X. Therefore, the goal of this research can be formulated in the main research question:

"How can a decision support system for transport route scheduling help Company X's planning personnel to create schedules faster and more efficient with lower routing costs?"

For the answer to this main research question, we already know on which areas improvements can be made. These areas are the 5 problems explained in Section 1.2.4. In this research, we search for the

degree in which we can reduce these problems or even completely solve them and how to measure possible improvements. To guide this process, we define the research questions in the next section.

1.4.2 Research Questions

To answer the main question, multiple smaller research question have to be answered. We introduce these questions per topic and give a description of the relevance of each question. Firstly, relevant information about the problems stated have to be gathered. This includes the functionality of the OOMHPA and how the planners use their own knowledge to plan routes. This helps us to determine how we will assess the improvements made by the decision support system. The following questions are relevant on this topic of problem context.

Problem Context (Chapter 2)

- What characterises the scheduling process at Company X?
  - To gain more insights into the different schedules Company X makes for their transport planning. Furthermore, to find which variables are important to incorporate into the solution and their relations.
- How do planning applications used in the scheduling process work at Company X?
  - To get a better understanding of how the current planning applications and OOMHPA handle the different processes regarding planning at Company X.
- How do planners work with the current applications and in the future with OOMHPA?
  - To define the differences for planners to work in the current and new situation and if the reasoning of planning decisions changes with the introduction of the new situation.

After defining the context of the problem, the scientific literature will be searched. This is to find all relevant information on the topic of decision support in the context of the problem. In this topic, the following questions are discussed.

Literature (Chapter 3)

- Which theoretical routing problem best describes the routing problem at Company X?
  - To gain insights into the work of other researchers into similar problem contexts and on how we have to bridge the gap between the context of Company X and the problem instances in theory.
- Which automated solution methods are applicable for the routing problem at Company X?
  - To retrieve knowledge on the possibilities of solving the problems most related to this problem.
- What can we learn from other research solving problem contexts similar to the context of Company X?
  - To gain insights into how other researchers have dealt with challenges similar to the problem presented in this research. Most attention will be paid to the translation of theoretical defined routing problems into practical, usable solutions.
- Which aspects need to be taken into account when designing a decision support system?
  - To determine what design aspects are working well in decision support and what should be avoided at all costs in decision support systems.

After gaining knowledge from the literature and from the current situation, we have to design the decision support system incorporating the characteristics of the problem context and the information from literature. The questions that need answers on this topic are the following:

Solution Design (Chapter 4)

- Which data from the OOMHPA can be applied into the solution design?

  - To provide information on what data from OOMHPA is available to use in the decision support system.

- Which other data is needed for decision support?
  - To explain which data is necessary to make a decision support system and also how this data can be obtained.

- What is the theoretical formulation of the scheduling problem at Company X?
  - To define the theoretical boundaries in which we have to design a solving method for the problem and to reduce the possible solution methods to design as a solution.

- How is a solving method for theoretical formulation designed in practice?
  - To explain the techniques used in the solving approach (or approaches) and how that results in the best solution given the assumptions, limitations & simplifications of the problem context.

With the solution design, the best possible configuration of the solution method within the decision support system has to be found. Experiments are used to find this configuration. With experiments, the following question will be answered.

Experiments (Chapter 5)

- How can we measure the performance of the created transport schedules?
  - To identify possible options and suitable metrics to compare the performance of the different schedules made by the designed solution.

- What parameter settings gives the best performance on the designed solution?
  - To find the best working settings for the different methods for the different support functions.

- How do the created schedules from the solution design compare to the existing schedules at Company X?
  - To compare and gain insight into the differences between manual created schedule and the schedules generated with our solution design.

Before drawing conclusions and providing recommendations to Company X, we want to provide them with insights in how to integrate the created solution within OOMHPA and how this solution can be used by personnel in different situations. Therefore, we investigate the following research questions on the topic of integration.

Implementation (Chapter 6)

- Which steps need to be taken to integrate the created solution into a decision support?
  - To determine the steps needed to create a functioning decision support from the created solution.

- How can planning personnel use the created solution to optimize the planning process and planning quality?
  - To describe possible practical usages of the solution and how this may achieve improvements on a practical planning solution rather than only on a theoretical solution.

- How does the solution designed for Company X compare to other routing problems?
  - To discuss the possible usages of the design in other contexts than that of the problem at Company X.

1.4.3 Scope

Before conducting research, the boundaries must be defined. In this section, we will list these boundaries and elaborate on how these boundaries affect our research. The main focus will be on the creation of a suitable scheduling algorithm which can be used within the scheduling process of Company X. Literature presents many solution methods to problems similar to the problem at Company X. However, this research will not be benchmarking or elaborately comparing all the different solution methods on the problem. Instead, the focus of this research will be on the most promising or relevant methods and applying them on the problem presented.

Another factor that limits the scope of the research is that we do not experiment in practice with the obtained results of our solution. This is because setting up valid experiments to test working with a decision support is too time-consuming for the time frame of this thesis. Therefore, in the implementation chapter (Chapter 6), we discuss possible use cases for the solution design and in the conclusion (Chapter 7) we elaborate further on possible research in practice with the solution design.

1.5 Outline of this thesis

The remainder of the thesis will have the following outline. Chapter 2 will discuss processes related to planning and context at Company X. In Chapter 3, scientific literature related to the problem will be presented. Chapter 4 continues with a solution design. In Chapter 5, the experiments are presented together with their results. Chapter 6 discusses the implementation of the solution. Chapter 7 presents the final conclusions, recommendations and future research.

2 Problem Context

In this chapter, we introduce the situation at Company X. To describe the processes in Company X in more detail to better understand how scheduling takes place at Company X. This is found in Section 2.1. Afterwards, we discuss the role of subcontractors (Section 2.2). Third, we describe how Operational Order Management Health Process Application (OOMHPA) will function and the eventual difference between current applications. Section 2.4 discusses which factors Company X has to take into account from the customers. Lastly, we discuss the details of the transport fleet at Company X.

2.1 Planning Process

The planning process at Company X is divided into several sub-schedules as discussed in Chapter 1, each with different purposes, activities, and results. All schedules are related and executed at different times. Figure 3 visualizes the workload and time spend on each planning. The times provided are the hours scheduled for each step. The hours are determined in such a way there is always sufficient time to create a finished schedule, even with unexpected events. This figure does not contain the subco planning, which is discussed in Section 2.1.3.

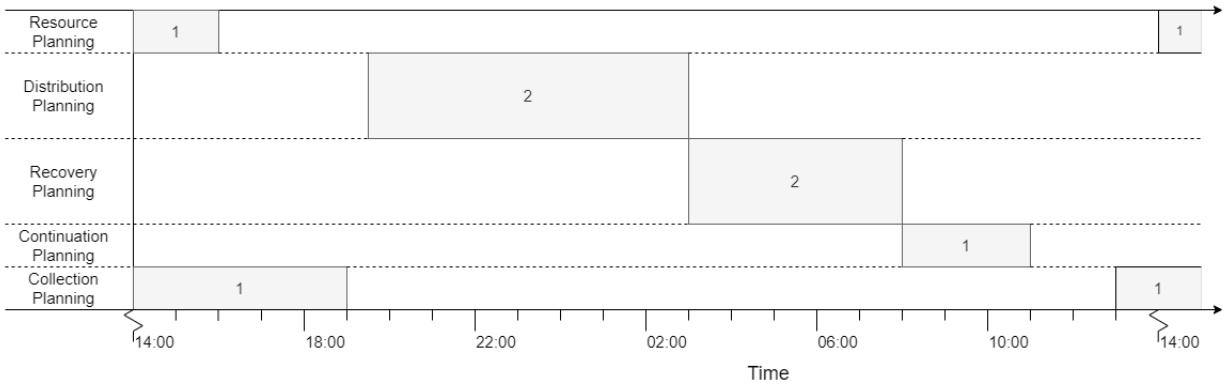


Figure 3: Planning process with times executed and the number of planners working on each planning

The start time of Figure 3 is 14:00, the start of the planning for the next day. The collection planning is still worked on for the current day, resulting in an overlap with the resource planning. The complete scheduling process for one day takes therefore from 14:00 until 19:00 the next day (29 hours).

2.1.1 Base Planning

The base planning is the foundation of the whole schedule created for the day. This schedule is not made every day by a planner. In fact, it is approximately the same as the planning years ago, except for changes in the client base. It contains customers locations which are linked to a specific vehicle called standard orders. These standard orders create each day a default customer list for every vehicle. This forms the basis for the distribution planning discussed later on and therefore is an important variable in the entire planning process. The great benefit is reduced time to create a schedule since a part of orders are already planned. Using a base planning for the distribution planning is possible because most customers of Company X have similar orders every few days. Therefore, it is important to have stability in the customer pool and in their order quantities, otherwise it can be contra-productive to use these base planning. Namely, the effectiveness is reduced if it only contains for example one or two out of the eight stops in a planning and if customers are removed and added regularly this base planning is not optimized to the changing environment. Since the base planning is an important factor for all other planning steps, we investigate in the remainder of this section the stability in orders at Company X and thus the validity of using standard orders in routes.

If there is a lot of fluctuation in volume, more mutations have to be made to create a distribution planning from the base planning. Diminishing the positive impact of using a base planning. Figure 4 shows that the number of colli transported by the own fleet decreases every year. One colli is one package, the size is variable, but at Company X this mostly resembles the size of a large box. However, this decrease is not compensated for by an increase in the volume of pallet transport (see Figure 5). This fluctuates more

than the trend in colli, however, no stable increase or decrease can be seen. The main reason for decrease in own transported colli is an increase in outsourced colli (see Figure 6). Fluctuating volumes throughout the year and a decrease in own colli volume jeopardizes the usefulness of the base planning. Therefore, it is important to investigate the effectiveness of (the current) standard orders for vehicles when comparing our solution design with the current schedules in Section 5.8.

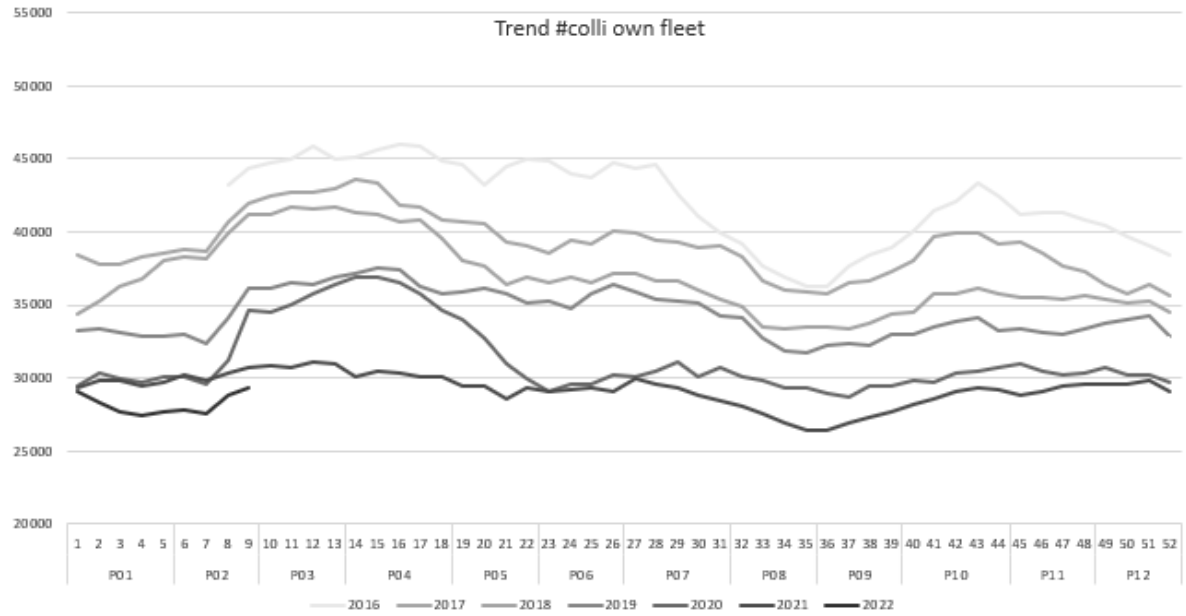


Figure 4: 8-week average trend development in percentage orders for subcontractors. Darker colours represent more recent years. The long term trend is a decrease in colli volume transported

2.1.2 Resource Planning

In the resource planning, resources are matched with the expected demand for the upcoming day. The resources that are matched are drivers and vehicles. The demands for routes are estimated on the expected volume of standard orders for a vehicle and eventual additional orders from other standard routes with are not executed that day. Most routes have a default vehicle and a default driver to execute the route. This planning is extended with irregular and direct orders. Direct orders are orders that must be collected and distributed in the same route. Most of the volumes are still unknown at this moment in time. Therefore, estimates are used internally, called placeholder volume. One of the goals of this planning is to get the starting times for the drivers the next day. Therefore, Company X creates this planning in the afternoon. This is done in the afternoon for the drivers, because it is more confidently for them to know their start time as early as possible. However, the later this schedule is made, the more precise the resources can be determined. The difficulty of this planning is to balance the costs of resources with the desired supply security.

The resource planning is created mostly on paper based on a default template. With the introduction of OOMHPA this will be done manually within a dedicated section of OOMHPA. This template contains the route that each vehicle/driver has to complete. This includes the information on start-time of the driver, the trailer for the vehicle (if applicable) and the default route which will be driven (Figure 7). Standard information can be extended with the following information: information about a second route, additional deliveries, direct orders, and other notes. This template is first changed digitally, then printed and then the changes are done by hand.

2.1.3 Subco Planning

The next step is determining the orders for outsourcing. This process is carried out with business logic based on the postcode of the delivery addresses (see Figure 8). Section 2.3 details this process. It results in a list of orders to be outsourced to subcontractors. This list is processed by planners into routes that will be driven by the different subcontractors. These routes are communicated to subcontractors so that

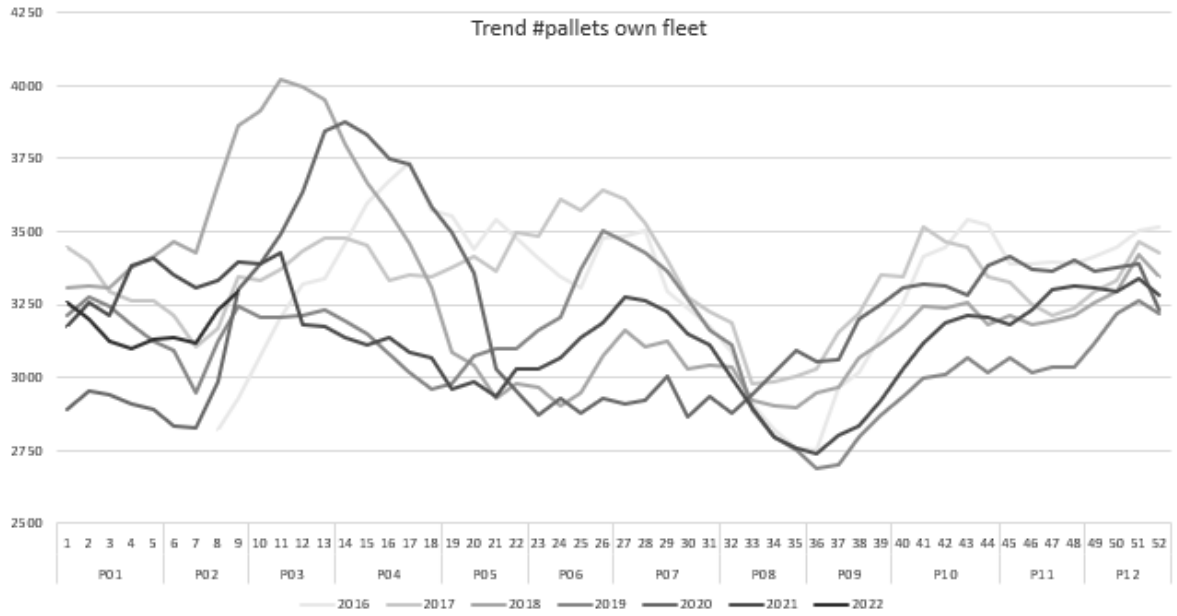


Figure 5: 8-week average trend development in number of pallets. Darker colours represent more recent years. No stable in/decrease in pallets volumes is noticeable.

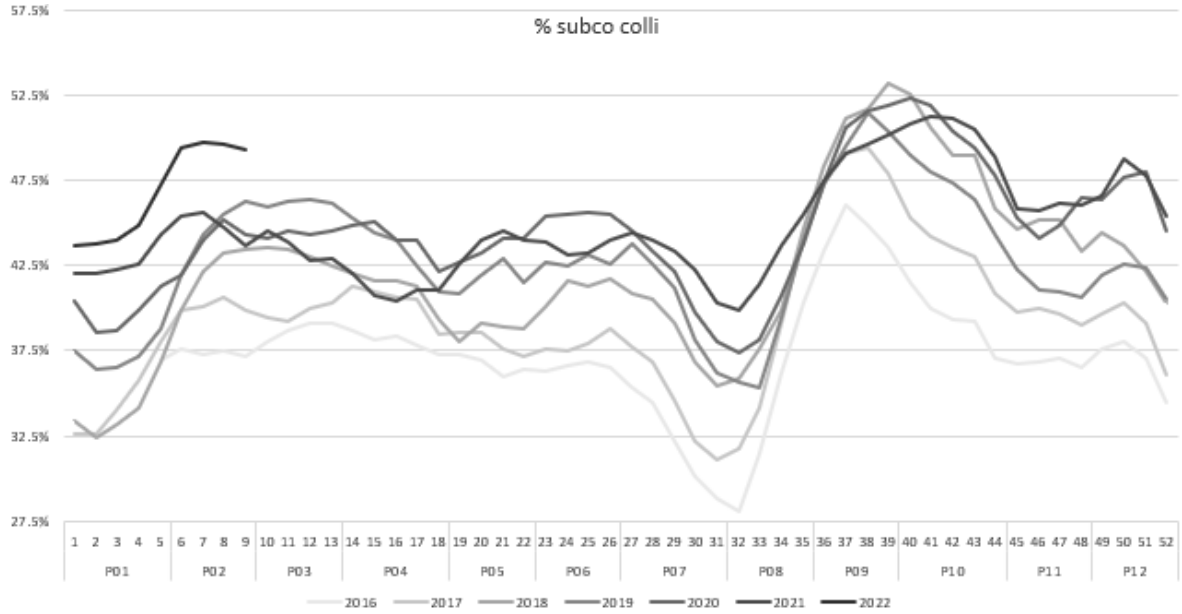


Figure 6: 8-week average trend development in percentage orders for subcontractors. Darker colours represent more recent years. An increase in percentage orders for subcontractors is visible.

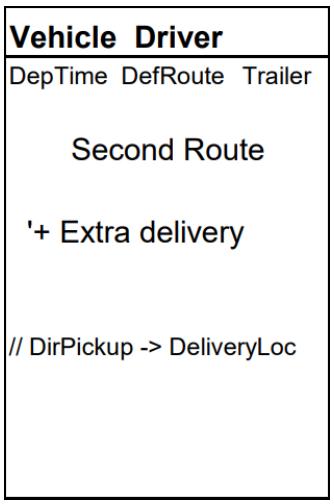


Figure 7: Default template for planning a vehicle in the resource planning

they know how late to pick up the deliverables. As can be seen in Figure 8, the subco also has a recovery planning step in the process (see Section 2.1.5).

The process of creating these subco routes is partly automated. Automatically, the orders are split for the different subcontractors. For some of the subcontractor which plan their routes themselves, these are directly send to them. For the other group, two additional steps are required to plan their routes. The first step is clustering of orders into groups suitable for a van. Afterwards, for each of these clusters, a route is automatically created by Logistic Dynamic Planning (LDP). Company X currently runs an experiment to automate the clustering process in LDP as well.

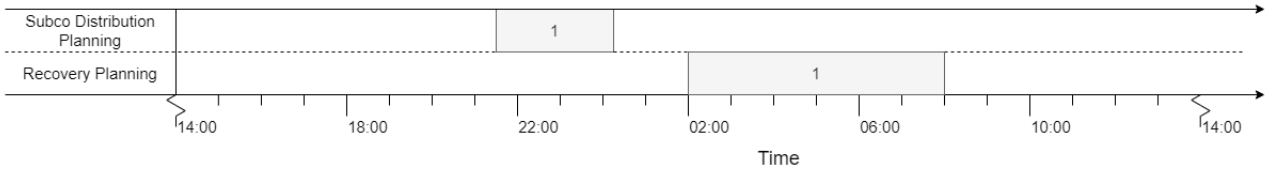


Figure 8: Time and workload visualization of planning steps related to subcontracted routes

For the routes of subco, Company X aims for at least 43 stops per vehicle because many stops are more cost-effective. Figure 9 shows the achieved number of stops per route in 2021. In the figure, we see that this target level is not always achieved. At the beginning of 2021, the number of stops is around the target. However, during the year, it deviates more from the target value, more often below than above the target. This illustrates that it is difficult to create cost-effective routes for subcontractors. More information on subcontractors is provided in Section 2.2.

2.1.4 Distribution Planning

After determining the outsourcing orders, the exact routes must be determined. In the time that has passed from the creation of the resource planning, more information about volumes has arrived. This may result, for example, in a violation of the capacity limit. Consequently, orders may be shifted to other routes.

The process of the distribution starts at 19:30, the deadline for customers to submit their orders to Company X for transport the next day. From this point onwards, planning personnel has to determine how the default routes match the submitted orders. The orders which can fit the default routes are placed in these routes. After this step, there may still be many unplanned orders. These are put into a non-driven route internally called the fish-basket. Personnel tries to empty this as much as possible, but this is not possible for all orders, those are handled in the continuation planning (Section 2.1.6). Some stops have orders which have to be delivered at different sub-location within the main address.

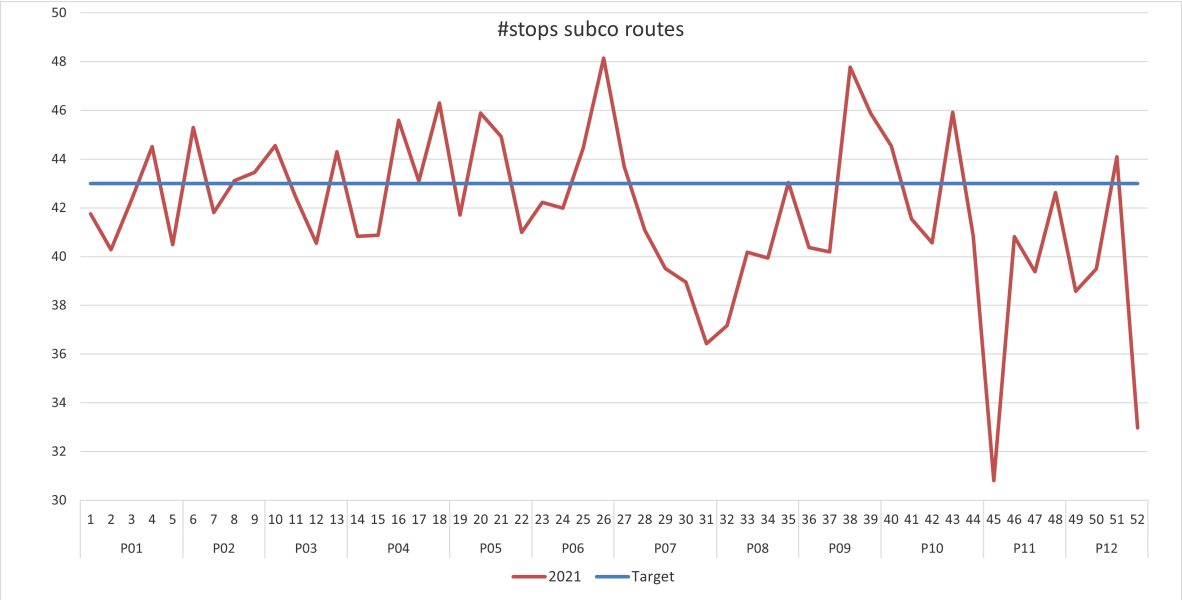


Figure 9: Average number of stops in subco routes per week. The target level of 43 stops per vehicle is not always achieved, resulting in less cost-effective routes.

An example of this is a university medical centre that has departments in multiple buildings. In those instances, the planner must also indicate the sub-location of delivery. This is done in sub-routes within a delivery. The results of the distribution planning are printed per order and sorted in route order so that warehouse personnel can load the vehicles in reverse order of delivery so that the service times are minimised.

One of the difficulties in this planning is the fact that planning the order in stops for vehicles is done without any support. So the planner has to order stops based on a list of addresses, postcodes and cities and their knowledge of geography. There is no support from a map which displays the locations of the addresses. The only information planners have for the stop sequence is their own experience of this specific route. If a route is planned suboptimal, it is difficult to change, since the orders are loaded in the route specific order. Therefore, more than only the order has to be unloaded to change the route of a vehicle. In most of the cases, this is more costly or even not possible to do. Consequently, changing routes is rare and unsatisfactory routes are driven even if sub-optimalities are discovered before execution.

2.1.5 Recovery Planning

The recovery planning is the planning in which unforeseen changes are dealt with. These changes entail before the start of a route. An example is driver illness. First, the planner searches for a substitute. If there is no other driver available, the orders have to be shifted as much as possible to other routes, otherwise they have to be (temporarily) delayed. The same applies, for example, to an unavailable vehicle. This step is perceived intensive by the personnel. This step starts at night and in practise overlaps with subco and distribution planning.

2.1.6 Continuation Planning

When the first orders are delivered to customers, the continuation planning starts. In the continuation planning, second routes are scheduled for vehicles. Some second routes already participated in the resource planning, but most of them are made with orders which first could not be scheduled in the normal planning. These orders were mostly not scheduled due to capacity limitations. Second routes may also contain default collection orders. Collection orders are orders to pick up goods from a location. Most collection order planning happens in the collection planning, explained in the next section. The goal of this planning is to minimise undelivered orders. The results of these efforts are shown in Figure 10. This figure shows that since 2017 no more than 2% of the orders do not end up in the vehicle.

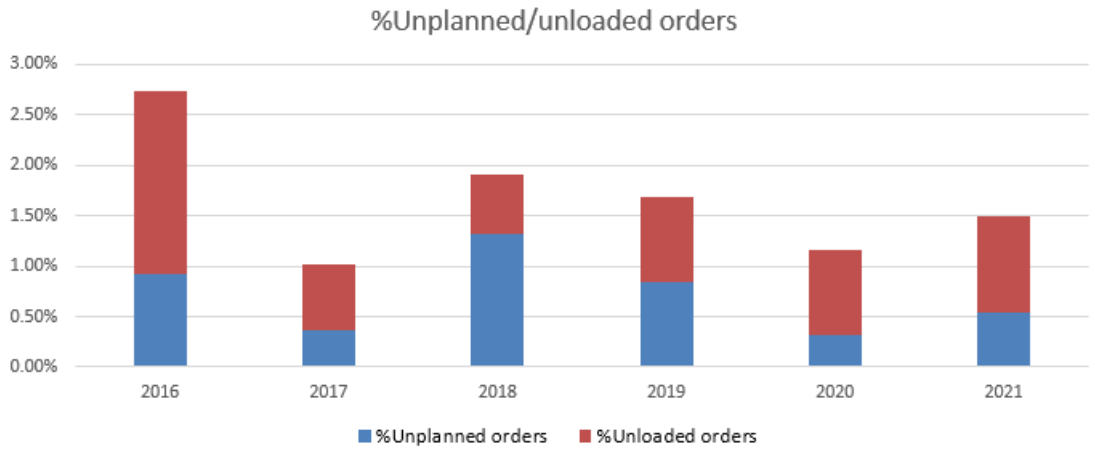


Figure 10: Average percentage of unplanned and unsorted orders

2.1.7 Collection Planning

The last step in the planning process is the collection planning. In the collection planning is determined which vehicle visits which pickup location in the afternoon. The process happens during the execution of delivery request, making this planning step, the only dynamic planning step. This depends on multiple factors of different importance. The most important is the closing time of the pickup location. Resulting in that these customers are prioritised when vehicles finish distribution. Afterwards, planners account mostly for the driver's drive time. A driver who worked less on that day is preferred for a pickup than a driver who already worked a long day. This is to reduce overtime for drivers, which is an important performance measure for schedules. A factor with a lower priority is distance travelled or additional time spent.

Currently, a supportive tool developed by one of the planners guides this planning. Before the planning department receives the final volumes, the planners have already assigned some capacity to certain pickup locations. This is done because planners know with high certainty that this volume always has to be collected, and this driver is always on time and close by to collect from this pickup location. Most collection planning occurs after the exact collection demand has been received. Then the planners assign the remaining capacity to the different pickup locations. The tool provides the planner with information on the available capacity of each vehicle, the unfulfilled pickup demand for each location and the current assignment of vehicles to locations.

The most time-consuming activity for this planning is to retrieve information on the available capacity of a vehicle. Almost always, not all the capacity is available for pickups. Therefore, planners have to contact the driver (by phone) to ask about the remaining available capacity for pickup orders in the vehicle; then the vehicles' remaining capacity is updated and can be used for the finite collection scheduling. This is done manually by the planner without decision support. Eventually, when the collection orders for the vehicles are determined, the drivers are either updated via the on-board computer or by the handheld.

2.2 Subcontractors

Subcontractors are transport companies which perform certain orders from Company X. The subcontracted orders are mostly small orders for pharmacies. Until many years ago, Company X performed these orders themselves. However, since then, performing these orders did not fit in the company strategy any more. Therefore, these orders have been subcontracted ever since. Subcontractors have contracts with Company X to perform orders in certain regions. In Table 1 all subcontractors are listed together with the region in which they operate, and we give an approximate size indication. The base tariff received by subcontractors depends on the distance between the region and the depot of Company X. These contracts result in that orders which better fit in the route of subcontractor A, but is located in region B, cannot be executed by subcontractor A. This is an example of the inflexibility in the subcontractor structure. However, as explained in Section 1.2, these contracts cannot be easily changed.

The own fleet of Company X is also visiting the different regions of the subcontractors. Therefore, Company X experimented in the past with performing some subcontractor orders with their own (larger) vehicles. These were orders closely located to locations already visited by our own vehicles. However, this was not a success for two reasons. Firstly, the small orders got lost in the large vehicles, resulting in long service times for these orders. Secondly, the sorting of small orders normally takes place around the same time the first vehicles depart. Therefore, for this experiment, the sorting procedure had to change. This created extra complexity and, thus, work for the sorting department. These factors make it impossible to solve the problem of placing subco orders in the distribution planning.

Table 1: Size and region indication of the different subcontractors of Company X

Subcontractor Number	**Region**	**Size Indication**
1	Antwerp	Large
2	West-Flanders	Medium
3	Central Belgium	Large
4	East-Flanders	Small
5	Northeast Limburg	Small
6	South Belgium	Medium
7	Liège	Medium

2.3 Planning Application

The OOMHPA is a project founded by Cape together with Company X and the IT team to improve their internal processes. The main goal of this project is to replace the current Enterprise Resource Planning (ERP) system with a new system, the OOMHPA. The main disadvantage of the current system is that the application is not configurable and cannot interact with other applications within the organisation. The OOMHPA is currently partly implemented for the scheduling of direct orders. In the upcoming period, this will be extended until the full replacement at the end of 2022. In Section 2.3.1, we will discuss how orders are handled in the OOMHPA.

2.3.1 Order Handling

One of the most important steps of OOMHPA is how it handles orders, especially regarding the scheduling activities. Namely, the logic applied to orders determines the route in which an order is planned. First, we discuss this for distribution orders and later for collection orders.

Distribution orders
In the OOMHPA there are multiple statuses for orders. That is, new, stopped, and ready for scheduling. New orders in OOMHPA arrive when a customer submits an order in the customer portal. On arrival, the order is checked for completeness of information. Orders are stopped on the following criteria: specific customer, no workday, submitted after submission deadline. Planners can change the status to ready for scheduling manually. When activated, business logic enriches the order information. This logic determines in steps the TUP. A TUP is a Dutch acronym for Transport Uitvoerende Partij (transport executing company). The definition of a TUP is broader than that of subcontractors. It also includes partner organisations, and even Company X itself. The logic for finding TUP difference per region in which the customer is located. If no TUP is found after all this logic, then it will be assigned to the default international transporter. The trip, driver and vehicle are yet to be determined, this happens in the resource planning.

When creating a resource planning, all delivery locations of possible orders are linked to trips, drivers, and vehicles. The resource planner can determine the starting time for each of the trips, and change drivers, vehicles with the availability of resources. Locations which are not yet connected to a trip need to be manually inserted into a trip. A location is by default connected to a trip if this is documented in the master data. This creates the following example overview for a trip; see Figure 11. In this figure, the delivery window for the different delivery locations is also shown.
The trips from the resource planner do not contain the real orders submitted from the customer portal. When the distribution planning is created, the real orders are used. These orders are then divided amongst the different overviews. For the own vehicles, the overview presents the orders per vehicle. The

Figure 11: Example of a trip within the resource planning in the OOMHPA

overview also presents a list of all vehicles and their number of stops and available capacity. The other schedules within the distribution planning overview, are the subco, recovery planning, partner planning basket and the no-trip planning basket. The subco contains all the orders and will transfer those to LDP to create schedules for the subcontractors. Orders that cannot be handled in the own schedule are moved to the recovery planning window. So, second trips can be filled with these orders. In the partner planning basket, orders are placed for TUPs with a partner agreement with Company X that are more suitable to fulfil these orders. Lastly, the no-trip planning basket contains orders that are not directly linked to one of the above-mentioned overviews. It may also be extended or emptied by the distribution planners by shifting orders to/from this overview to/from, for example, the own first planning.

Collection orders
Collection orders will work very similarly to distribution orders in the system. At least for the structural collection orders. These are added into the resource planning. However, ad hoc collection orders will be placed in the recovery planning and will be moved into trips as much as possible at a later time.

Direct orders
Direct orders work similar in the OOMHPA as the other order types. However, planners must take into account the place of the direct order into the route. Especially with smaller direct orders, there are a lot of possible positions in trips for the pickup as well as for the delivery.

2.4 Order Details

The products Company X transports are medical supplies for mainly hospitals and pharmacies. Consequently, there are high demands from customers on the delivery quality of their orders. In this section, we discuss the most relevant details of orders.

One of the restrictions related to orders is the ADR. The ADR limits the number of hazardous substances that can be transported in a trip. In practise, only a small fraction of orders has ADR points. Consequently, this also has only a small influence on the planning. ADR orders are treated the same as normal orders in the planning, as explained in Section 2.3. However, trips containing orders with ADR restrictions are marked internally for review if the ADR limits are not exceeded. This can be checked relative easily since each customer submits the ADR point quantity together with the order information. Subsequently, the distribution planner has to only check if the summation per trip does not exceed the limit. When the limit is exceeded, the planner has to plan a driver with certificates to still comply with the ADR regulations.

A second important characteristic is the temperature range in which the order must remain from the delivery from the supplier to the customer. Company X differentiates 4 categories, which are shown in Table 2. For every order, it is known upfront in which category it has to be transported. Therefore, the capacity within a vehicle can be arranged. More details on this are given in Section 2.5. Orders without a temperature range can also be placed in the regular or cooled temperature zone. Whereas this is not true the other way around.

Another detail in the planning is the delivery (or pickup) location of an order. Section 2.1.7 mentioned

Table 2: Temperature ranges of orders Company X can transport

Full Name	**Abbreviation**	**Temperature Range**
Good Distribution Practice Standard	GDPS	No range
Good Distribution Practice Regular	GDPR	15-25 °C
Good Distribution Practice Cooled	GDPC	2-8 °C
Good Distribution Practice Frozen	GDPF	-30 °C

that the closing time of a pick-up location is the most important factor in the collection planning. This closing time is basically a component of the time window. All locations have these time windows, but some distribution locations also have a fixed delivery time agreement. This is the case for some larger customers, who want the orders to arrive structurally at the same time. To guarantee this, Company X plans these orders mostly as the first stop in a route.

For locations, there is also another limitation that is related to the physical characteristics of the location. In practice, this means that certain vehicles cannot reach a certain location because, for example, the streets are too small for the largest vehicle. This information is known by the default driver of the route and by the most experienced planners. Some location information is only known by the default driver of the route, such as the contact person at the location and other procedural action at the customer. Therefore, the planner may inform the driver upfront, but also accounts additional service time in for a new driver on a route.

2.5 Fleet Details

The fleet of Company X is important to ensure the quality of transports, having 4 vehicle types. Table 3 summarises the types of vehicles and their characteristics. Vehicle type 7 is the most available and versatile vehicle, a large truck which can be either used with a separate cool unit or can be used with two temperature zones. The type 8 truck is a smaller truck, suitable for the more restricted customer locations. A disadvantage is the reduced capacity and the absence of a cool unit. Truck type 9 is the largest vehicle with the most capacity, however it misses a separate cooling unit. Type 10 is the vehicle with the largest capacity. This is because it has a trailer capacity on top of the standard front load capacity. However, the trailer capacity is assigned to orders of one or two wholesalers. Since the trailer has to be left behind in order to access the front load. However, orders from other customers can be added to the trailer as long as their load is of the same type and only contains load of that type. When this happens, these orders have to be delivered before the orders of the default wholesaler(s).

Table 3: Characteristics of the vehicle types within the fleet

Vehicle Type	**Number of vehicles**	**Pallet Places**	**Cooling Unit**	**Compartment separation**
7	23	19	Possible	Possible
8	6	15	No	Yes
9	7	32	No	Yes
10	9	16+19 (35)	Yes	Yes, max 16 capacity

The capacity is displayed in a unit internally called “pallet equivalent”. Orders may be submitted per colli or per pallets. However, colli are combined into pallets for transport. Firstly, colli load is fitted as much as possible on pallets of the loadtype for the same customer. On average, additional 10 colli fit on each submitted pallet. For the remaining colli load, new pallets have to be created. In general, creating new pallets depends on the type of load. Cooled (GDPC and GDPF) loads are only combined into pallets of their own type. The other two categories, which are the majority of the load in general, are combined into pallets of the regular (GDPR) type. How much colli fits on a pallet depends on the customer type. On average 50 colli from hospitals and pharmacies orders fit on a pallet, whilst for wholesalers this is only 40. This procedure leads to a “pallet equivalent” for each order.

If a truck has compartment separation,, the vehicle can comply with two temperature zones instead of one. This is beneficial since a lot of customers require cooled as well as regular load, which can both

be served by this vehicle. When compartment separation is applied to a vehicle, both zones are still accessible, therefore, not restricting the vehicle's routing.

The described fleet of Company X does not have the capacity and suitable vehicle types to carry out the subco routes themselves. These routes are done with vans, which are currently not in the fleet. In addition, new personnel has to be contracted to drive those vehicles. Whether this is achievable and beneficial is beyond the scope of this research.

2.6 Conclusion

In this chapter, we described planning operations at Company X in detail and gained the insight that base planning becomes more complex due to a decrease in own colli transport per year. Consequently, the usefulness of the variable standard orders should be investigated. The distribution planning stands out as the planning component which contributes the largest to the core problem (see Section 1.2.4). Firstly, it is the most experience dependent process due to the lack of support. Secondly, the same planning personnel in the night has to deal with recovery planning when faced with unforeseen events. Consequently, the effectiveness of recovery planning is reduced because of the previously performed work. Lastly, the distribution planning is the most labour-intensive of all planning steps, it requires the most hours to create and therefore has potentially the largest scheduling time savings. When scheduling the distribution planning, personnel has to minimize driver overtime, total drive time and travel distance without any indication on these measures, while maximizing the number of satisfied requests within the time window of the customer.

The current applications which planners work with are not integrated which each other. Most planning step use a different application to administrate their created route, shifted order, new stop etc. Some contain a large manual or even sometimes analogue component. These factors together makes working with the different planning steps even more experience dependent. However, this becomes less of a problem with the introduction of OOMHPA.

The main difference in working with OOMHPA over the old application environment is that OOMHPA will contain information in one system. This mainly helps the accessibility for planners of general information on for example customer time windows, but it does not support specific scheduling information such as travel times of vehicle route or actual or estimates loads of a vehicle. Therefore, in the core, the planning process will not change drastically in the new situation compared to the old situation.

In general, we concluded that our focus should be on supporting the planning personnel in creating the distribution planning and helping them with changes necessary due to unforeseen events. The solution designed should be able to handle distribution orders, structural collection orders and direct orders. This will be discussed further in Chapter 4. Besides incorporating different constraints, another challenge is to inform the planner with accurate performance measures and relevant alternatives when creating or changing the planning.

3 Literature review

This chapter provides an introduction to the literature related to this research. In Section 3.1 we discuss the Vehicle Routing Problem (VRP) and the relevant variants. Subsequently, Section 3.2 discusses the different approaches in the literature to solve these types of problems. Lastly, Section 3.3 reviews the literature on decision support in the field of transport scheduling and how to optimally present information from a decision support in a visualisation to the end-user.

3.1 Vehicle Routing Problems

The VRP was first defined by Dantzig and Ramser in 1959. The paper discusses the problem of finding an optimum routing of a fleet of petrol delivery trucks from a terminal to supply service stations. It marked the beginning of a still growing research field (Tan and Yeh, 2021). In this section, we first discuss formulation types of VRP, then the two problem characteristics are discussed in Sections 3.1.2 & 3.1.3. Lastly, we discuss multiple relevant variants in Sections 3.1.4-3.1.11. This is to find out which formulation suits the problem at Company X, which is discussed in Section 3.1.13.

3.1.1 Formulation

The VRP is most commonly modelled as Mixed Integer Linear Programs (MILP) (Sidorov and Morozov, 2021). The most widely used version is the vehicle flow formulation. It is based on binary variables associated, which indicates whether arcs between locations are travelled. In general, this is more intuitive and leads to a compact model. This formulation is mostly used in literature. Mostly, the two or three index variants are used. The difference between these variants is that if the problem has a homogeneous fleet, then a two-index formulation is sufficient (Sbai et al., 2022), the two indexes represents the two locations of an arc. For the heterogeneous fleet, a three-index formulation is necessary. This to indicate the specific vehicle with the third index, as presented by Li et al. (2021). Another formulation is the set partitioning formulation originally proposed by Baldacci, Christofides, and Mingozzi in 2008. It assigns a binary variable to each feasible route. Balinski and Quandt (2008) shows a set partitioning formulation for a similar problem as in (Sbai et al., 2022).

3.1.2 Deterministic/Stochastic

A VRP has to be either deterministic or stochastic. However, the amount of stochasticity varies in different stochastic VRPs. In a deterministic VRP, inputs do not contain any probabilistic values. Stochastic input can be in multiple categories. Most common are stochasticity in customers (demand) and in travel or service times. An example of the latter is presented by Gutierrez et al. (2018), who developed a solution method for the latter case. Berhan et al. (2014) identify four models for the Stochastic Vehicle Routing Problem (SVRP) of which the Chance Constrained Program (CCP) & Stochastic Program with Recourse (SPR) are most common (Oyola et al., 2018). In the CCP the probability of failing is minimized, but failing costs are not accounted for. Within a VRP context, this could be that the probability of failing a time window is minimized, but no costs for failing are incorporated in the solution value. In the SPR failure is allowable, but it needs to have a recourse policy which describes the actions and costs to repair the solution. For example, failing a time window is more allowed, but it comes with a penalty to the solution value. Gendreau et al. (1996) summarizes their difference as follows: "SPRs are typically more difficult to solve than are CCPs, but their objective function is more meaningful". When comparing stochastic VRPs with deterministic VRPs, the stochastic problems are a more accurate representation of reality but are more difficult to model, solve and interpret.

3.1.3 Dynamic/Static

The general VRP is considered a static VRP. The problem inputs are known upfront and do not change thereafter. This is different in the Dynamic Vehicle Routing Problem (DVRP). A VRP is dynamic if the input on the problem is received and updated concurrently with the determination of the route (Psaraftis, 1988). For most of the DVRPs, the fact that some transport request are arriving during the execution of routes. An example of this is Pickup and Delivery Problem (PDP) (Section 3.1.12) of Gendreau et al. (2006). The largest implication is not knowing all customers when determining the routes, resulting in that decision on which vehicle takes an appearing customer have to be in real-time. This real-time decision-making also holds for other dynamic elements such as vehicle breakdowns (Psaraftis et al., 2015).

3.1.4 Backhauls

The first variant we discuss is the Vehicle Routing Problem Backhauls (VRPB). The standard VRP considers online linehaul customers, e.g. customer with a delivery request, while in the VRPB also contains collection request (backhauls). The first mention of the term is made by (Goetschalckx and Jacobs-Blecha, 1989). Parragh et al. (2008b) distinguishes four separate classes within the VRPB with goods transport. Other researchers describe similar variants differently, therefore we summarised the variant names and explanation in Table 4. In the formulation of the VRPB, the customer set is split between linehaul and backhaul customers, and constraints are too, if necessary. Çagrı Koç and Laporte (2018) presents both an ILP and a set partitioning formulation.

Table 4: Names and explanations of different VRPB variants discussed in literature.

VRPB variant	**Explanation**
VRP with Clustered Backhauls (VRPCB)	All linehaul customers are served before backhauls customers
VRP with Mixed linehauls and Backhauls (VRPMB) Pickup and Delivery Problem (PDP) Mixed VRP with Backhauls (MVRPB) VRP with Backhauls with Mixed load (VRPBM)	No restriction in the order to serve backhaul and linehaul customers.
Single Vehicle Routing Problem with Pickups and Deliveries (SVRPPD) VRP with Divisible Delivery and Pickup (VRPDDP)	Customers can be both linehaul and backhaul customers. If customers are both, its not necessary, in this subclass, to visit only once.
VRP with Pickup and Delivery (simultaneous VRPPD) VRP with Simultaneous Distribution and Collection (VRPSDC) VRP with Simultaneous Delivery and Pickup (VRPSDP)	Customers can be both linehaul and backhaul customers. If customers are both, they are still only visited once.

3.1.5 Time Windows

One of the most used variants of the VRP is the Vehicle Routing Problem Time Windows (VRPTW). Pullen and Webb (1967) were among the first to work on this variant. In this variant, customers have a time in which they can be served, a time window. Therefore, for each location a start, end and service time has to be given. Between each location, a travel time is necessary and, for each vehicle, a start time. With this information, the arrival time can be calculated given the previous stops. The modelling constraints for the VRPTW can be hard or soft. Hard time windows do not allow service outside the given time window, while soft time windows penalize service outside the time window. The taxonomy of Gutiérrez-Sanchez and Rocha-Medina (2022) presents four sub variants of VRPTW, which are given in Table 5.

Table 5: Different VRPTW variants discussed in Gutiérrez-Sanchez and Rocha-Medina (2022).

VRPTW variant	**Explanation**
VRP with Multiple Time Windows (VRPMTW)	Customer may have multiple time windows in which they can be served.
VRP with Time Deadlines (VRPTD)	Customers have to be served before a certain deadline.
(Multi) VRP with Soft Time Windows ((M)VRPSTW)	Customers are allowed to be served outside the time window with a penalty on the objective.
Vehicle Routing and Truck Driver Scheduling Problem (VRTDSP)	A VRPTW variant which incorporates driver rest periods.

3.1.6 Subcontracting & Outsourcing

Subcontracting or outsourcing is not extensively studied in the literature. Most problems do not deal with subcontracting or outsourcing. There are two categories to extinguish. The first is subcontracting orders, and the second is the use of outsourced vehicles. The first subclass can be modelled by creating additional decision variables to indicate whether the demand is satisfied by own vehicles or subcontractors (Chu, 2005). The second subclass is dealt with by increasing the size of the fleet with outsourced vehicles and correctly representing their cost compared to the own vehicles (Moon et al., 2012).

3.1.7 Time-Dependent

The Time Dependent Vehicle Routing Problem (TDVRP) is a variant of the VRP in which travel times between locations depend not only on distance, but also on time of day (Malandraki and Daskin, 1992). The first studies in this field use piecewise linear function to define travel times between two points (Figure 12).

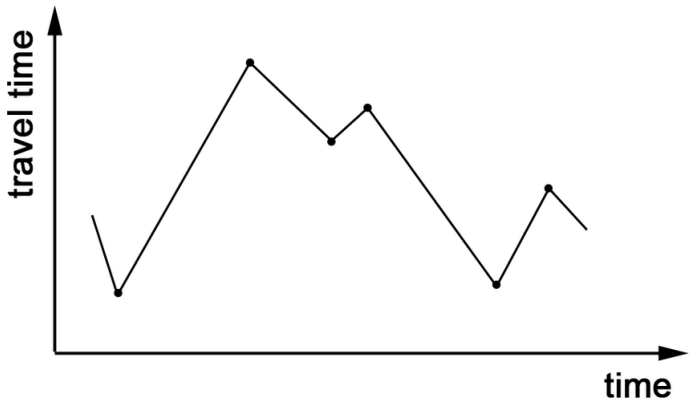


Figure 12: Example of a piecewise linear function for a TDVRP

Ichoua et al. (2003) presents another view on that makes the travel time dependent on the travel speed. This more accurately presents the travel time, since this representation ensures that the first departing vehicle from point i, always arrives first as well at point j. More recent studies incorporate historical data in their models (Jula et al., 2008) (Woensel et al., 2008) (Caric and Fosin, 2020). All these approaches result in some form of speed profiles for distances between locations. These approaches differ in complexity to find these speed profiles.

3.1.8 Multi-Depot

The Multi-Depot Vehicle Routing Problem (MDVRP) is a VRP variant in which the customer can be served from multiple depots. Montoya-Torres et al. (2015) presents a 3-index formulation of this problem. Important to note for the formulation presented is the fact that all customers can be served from all depots. To the best of our knowledge, no research considers the limitation that certain customers can only be served from a specific depot(s).

3.1.9 Multi-Trip

The Multi-Trip Vehicle Routing Problem (MTVRP) was firstly defined by Fleischmann (1990) as the VRP with multiple use of vehicles. Most of the different names for this problem are given in Cattaruzza1 et al. (2016). This paper also presents multiple formulations from different papers on this problem. It ranges from 4-index formulations to 2-index formulations. As the name of the problem class suggests, it allows vehicles to make more than one trip a day. In the standard VRP, a trip is exactly the same as a route, but in MTVRP a route can have multiple trips (see Figure 13).

3.1.10 Multi Compartment

The MCVRP is a VRP variant which models the availability of two separate compartments within the same vehicle. The most used application of these models is within petroleum or food/grocery transport. Chajakis and Guignard (2003) are among the first to present optimisation models for this problem

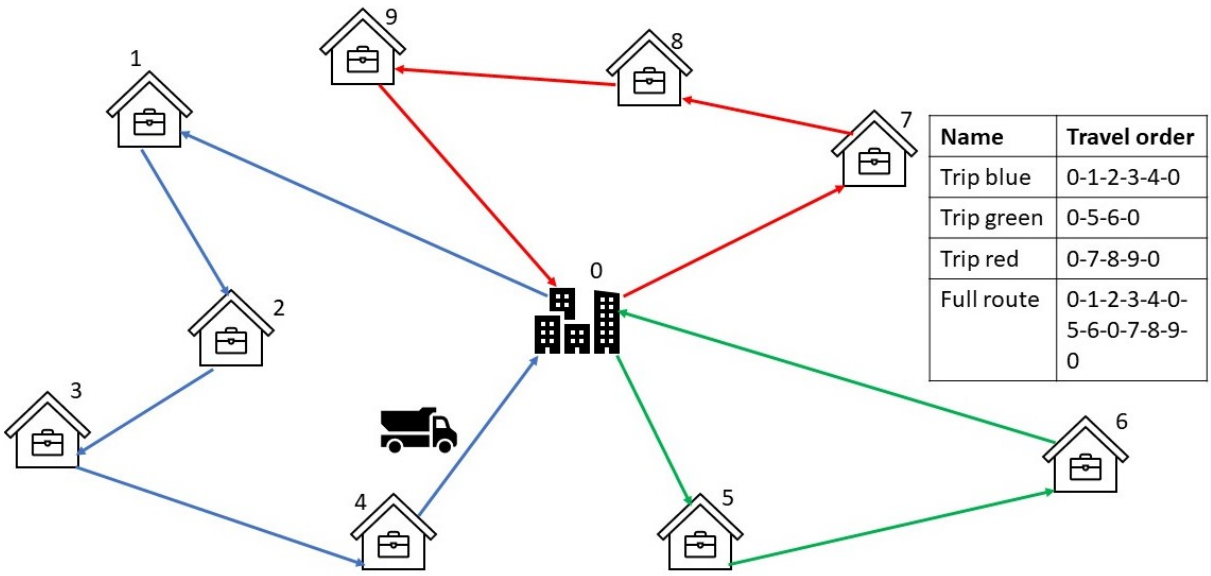


Figure 13: Visualization of the difference between trip and route in a MTVRP

class. The formulation for these problem classes extends the capacity constraints restricting the load of compartment i to its size and the total size of all compartment types to the vehicle size. Other researchers have dealt with different aspects that arise when working with compartments, such as costs or collection orders (Hübner and Ostermeier, 2018) (Muyldermans and Pang, 2010).

3.1.11 Open VRP

The Open Vehicle Routing Problem (OVRP) variant does not have the limitation that the vehicle has to return to the depot. The consequence could be in the the drawn instance of Figure 13 that arc 9-0 does not have to be driven. Not returning to depot is in many cases not an option for companies. This fact, combined with the fact that the OVRP is easily created from the VRP and has a similar complexity, results in the fact that this sub-problem is not studied much in the literature.

3.1.12 Pickup and Delivery Problems

The last VRP category that we discuss is the Pickup and Delivery Problem (PDP). It is closely related to the VRPB, with the difference that the goods for delivery are yet to be picked up. Parragh et al. (2008a) presents multiple subclasses within the PDP with multiple vehicles. These are shown in Table 6.

Table 6: Different PDP subclasses discussed in Parragh et al. (2008a)

PDP subclass	**Explanation**
Pickup Delivery VRP (PDVRP)	Every unit picked up can be used to serve every customers demand.
Pickup and Delivery Problem (PDP)	Every pickup point is paired with a customer delivery location
Dial-a-ride Problem (DARP)	An extension of the PDP which incorporates max transport time for the transport of persons.

The formulation of the PDP does not differ much from the formulation of the VRPB given in part i of Parragh et al.. Only for the paired cases there are two additional constraints needed, namely that the same vehicle has to do the pickup and the delivery, and that pickup occurs before delivery.

3.1.13 Relevant problem formulation

In previous sections, many variants of the VRP are discussed. In this section, we use this information to conclude which VRP problem definition is applicable for the case of Company X. Foremost, the

distribution planning is a deterministic and static problem, since the requirements for a dynamic and/or stochastic problem are not met. The PDP is necessary over a VRPB due to direct orders. Therefore, the base problem is PDP.

Not all the variants discussed are applicable to the situation in Company X. Table 7 discusses all variants and explain the reasons why these variants are applicable to the problem context. Given the problem context and the definitions from literature, we define the problem as a deterministic, static Multi-Trip Multi-Compartment Pickup and Delivery Problem Time Windows (MTMCPDPTW). Deterministic, since stochastic is only present in unknown travel and service times. The increased complexity when modelling these variables stochastically would not outweigh the relative small increase in performance, since those variables can be estimated reasonably accurate. Static modelling is not necessary since order inputs are known upfront. The MTMCPDPTW is to the best of our knowledge not yet defined in literature (see Table 8). None of the sources solve a problem with multiple trips, multiple compartments and time windows in a PDP setting. The full formulation of the MTMCPDPTW is provided in Chapter 4.

Table 7: Relevance of different VRP variants in the problem context of Company X

VRP variant	**Relevant**	**Explanation**
Time Windows	Yes	Company X has to deal with customer which have an agreement to be delivered early in the morning, as well as customers have a closing time of the locations.
Subcontracting/ Outsourcing	No	Although subcontracting is part of the whole scheduling process at Company X it is separated and therefore a separate problem to solve.
Time-Dependent	Maybe	Time dependent travel times improves the accuracy travel times and therefore the routes. This is especially helpful when having small time windows and severe consequences of not meeting time windows. In addition, TDVRP models are more computational demanding. Therefore, this is less relevant then other variants and will not have a priority.
Multi-Depot	Maybe	MDVRP is currently not applicable since the problem only covers 1 depot. However, there is a desire to extend operations to other depots. Therefore, a multi-depot variant is nice addition, but not a necessity.
Multi-Trip	Yes	Multi-trips are a crucial aspect (see Section 2.1.6) within the problem context.
Multi-Compartment	Yes	Modelling multiple compartments will better represent the fleet characteristics and the customer demand of certain orders.
Open VRP	No	Vehicles of Company X are required at depot at the end of the day, therefore, open VRP is not relevant.

3.2 Solving methods

Now that we have defined the problem instances presented in the literature, we investigate the ways in which the literature has solved these different problem instances. Literature presents three general solving methods: exact (i), heuristics (ii) and meta-heuristics (iii). Exact methods are solution methods that generate the optimal solutions for the problems. However, all variants of the VRP are NP-hard problems (Lenstra and Kan, 1981). Consequently, exact algorithms are only suitable for relatively small and non-complex VRP instances. The case of Company X is not small, rather complex and fast solutions are preferable. For the recovery planning, a suggested solution should not take more than a few minutes

Table 8: Problem characteristics from (indirectly) cited literature in Sections 3.1.4-3.1.12

(a) Direct referenced sources

Literature	**VRP or PDP**	**MT**	**MC**	**TW**
Goetschalckx and Jacobs-Blecha (1989)	VRP			
Çagrı Koç and Laporte (2018)	VRP			
Pullen and Webb (1967)	VRP			X
Chu (2005)	VRP			
Moon et al. (2012)	VRP			X
Malandraki and Daskin (1992)	VRP			
Ichoua et al. (2003)	VRP			
Jula et al. (2008)	VRP			
Caric and Fosin (2020)	VRP			X
Woensel et al. (2008)	VRP			
Fleischmann (1990)	VRP	X		X
Chajakis and Guignard (2003)	VRP		X	
Hübner and Ostermeier (2018)	VRP		X	
Muyldermans and Pang (2010)	VRP		X	

(b) Number of sources (indirectly) referenced per problem per Taxonomy

Taxonomy	**VRP or PDP**	**MT**	**MC**	**TW**	**MT MC**	**MT TW**	**MC TW**	**MT MC TW**
Parragh et al. (2008b)	VRP	0	0	11	0	0	0	0
Gutiérrez-Sanchez and Rocha-Medina (2022)	VRP	1	0	23	0	2	0	0
Cattaruzza1 et al. (2016)	VRP	1	0	0	0	8	0	0
Parragh et al. (2008a)	PDP	0	0	26	0	0	0	0
Montoya-Torres et al. (2015)	VRP	1	0	33	0	0	0	0
	PDP	0	0	9	0	0	0	0

to be generated. For distribution planning, this is about an hour. Therefore, we do not focus on exact methods. Heuristics creates solutions based on simpler procedures. The literature on this topic is discussed in Section 3.2.1. Lastly, Section 3.2.2 discuss meta-heuristics which are algorithmic solving procedures with heuristics.

3.2.1 Heuristics

Heuristics for VRP have multiple applications in practice and come in multiple forms. The first method is constructive heuristics. Constructive heuristics create feasible solutions from scratch. These procedures are the fastest solution methods available, but do not yield the best solutions (Hoos and Stützle, 2005). Therefore, in practise, heuristics are used as starting solutions in metaheuristics (see Section 3.2.2) or combined with improvement heuristics to improve their quality. The more complex procedures which combine constructive and improvement heuristics are covered at the end of this section.

Constructive Heuristics
In construction heuristics, customers are inserted in routes (which may be empty) on a specific criterion or multiple criteria. One of the most intuitive examples is the nearest-neighbour insertion. In this heuristic, the customer added to the route is chosen based on the shortest distance to a vehicle. When this customer is chosen, the stop is added to the vehicle and the customer is removed from the unvisited customer list. This process is repeated until all customers are served. When dealing with more complex VRP, the nearest customer still has to apply additional constraints, such as time windows or vehicle capacity; otherwise the solution will be infeasible. Another basic construction heuristic is the cheapest insertion, which inserts the customer with the smallest increase in objective value into the route.

Improvement Heuristics
Improvement heuristics are used to improve existing solutions for problems. Improvement heuristics can be applied within a vehicle route or between the routes of different vehicles. The most basic improvement heuristics are the swap and move procedure. With a swap two stops are exchanged, this can be within a route or between routes. The move procedure removes a stop from a route and inserts it into another route. The criteria to determine which stops to move or swap are often greedy or random. Greedy means that the procedure will choose the option that generates the largest possible improvement, for example, removing the most expensive visit from the cheapest route. However, this will eventually lead to a local optimum, which is typically not the optimal solution (global optimum). Random improvement heuristics do not have this problem, but will make more often worse solutions than it started with. Another well-known improvement heuristic for VRPs is the k-opt heuristic. In this heuristic, the k-connections of a route are removed and then restored in the optimal configuration. Figure 14 illustrates how a 2-opt works in an improvement heuristic.

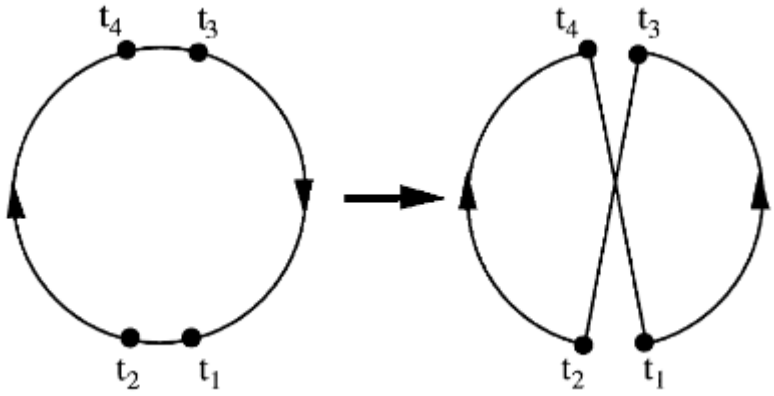


Figure 14: Illustration of the 2-opt in an improvement heuristic by Helsgaun (2000)

Heuristic algorithms
Heuristic algorithms focus on systematically finding acceptable solutions within a limited, predetermined number of iterations (Tan and Yeh, 2021). One of the first heuristic algorithms is the Clarke and Wright savings algorithm (Clarke and Wright, 1964). Heuristic algorithms create a first (feasible) starting solution

which afterwards will be improved or made feasible. The savings algorithm works with the following steps. In the first step, routes will be created from depot to every location and back to depot. Afterwards, the savings will be calculated for combining delivery location into one route. Subsequently, the list is sorted in descending order. The second step consists of merging routes with each other from the saving lists, providing that meet the constraints of the problem.

Similar to the savings algorithm are the two-phase algorithms; Cluster-first, Route-second, and Route-first, Cluster-second. The most used version of Cluster-first, Route-second is the sweep algorithm (Gillett and Miller, 1974). In this algorithm, the clusters are made by sweep around the depot. Every customer which fits in the vehicle is added to the cluster, otherwise a new cluster will start from this customer onwards. When the clusters are created, a route will be constructed by solving the remaining Traveling Salesman Problem (TSP) separately. The Route-first, Cluster-second approach starts by solving the TSP version of the problem first. Afterwards, clusters are created by breaking up the tour when the next customer does not fit the vehicle capacity any more. These versions of 2-phase heuristic algorithms work only for the basic VRP. Other researchers, for example, Prins (2002), created heuristic algorithms for more complex VRP instances.

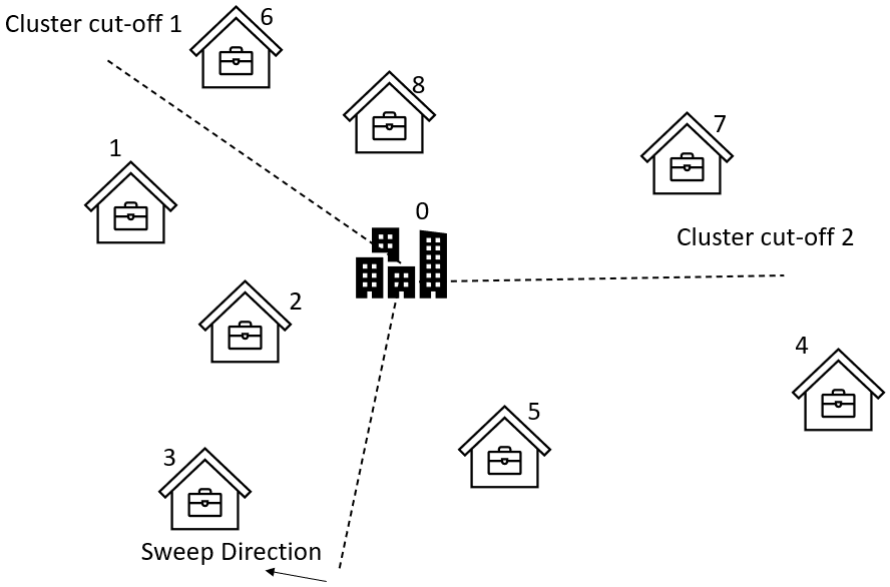


Figure 15: Example of the Sweep Algorithm

3.2.2 Metaheuristics

Meta-heuristics have in the literature multiple definitions. Voß et al. (2012) summarises these definitions as follows:"Metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions". The family of metaheuristics is large, therefore, in this research we limit ourselves to the most studied ones. Those are two categories: population or local search methods (Lin et al., 2014).

Population Search
Population search algorithms are applied in many areas of optimisation problems. These algorithms successfully update their population by a new, better population (Innocente, 2006). Another advantages of this method is that being trapped in local optima is less likely due to the parallel exploration of the population. Most of the population search methods find their origin in processes in nature. In VRP are the most common applied: Genetic Algorithm (GA), Ant Colony Optimization algorithm (ACO), Particle Swarm Optimization (PSO) (Tan and Yeh, 2021). The GA is based on Darwin's evolution theory. In the VRP context, this means that a child (a new solution) consist of a mixture of the two parent solutions. This is done a predetermined number of generations (iterations). Wester (1993) elaborately presents the procedure of on solving the VRP with a GA. Yusuf et al. (2014) present a clear overview on how a GA can be applied for solving a relative complex routing problem. The ACO is based on the food search

from ants in a colony. Ants choose their path based on a chemical substance called pheromone secreted from the ants to go through previously. For VRP, this means that ants pick a vehicle to operate and afterwards the nodes to visit. The more an arc is traversed (more pheromone) the higher probability an ant uses this arcs in their tour. Ky Phuc and Phuong Thao (2021) discusses the working of the ACO in more details. In recent studies on ACO in VRP adapted or improved versions of ACO are presented (Jia et al., 2021) (Li et al., 2019). The PSO paradigm originated from Kennedy and Eberhart in 1995. It was originally designed for simulating social behaviour for bird flocks or fish schools (Kennedy and Eberhart, 1997). PSO performs searching via a swarm (population) of particles (candidate solutions) that update per iteration (Zhang et al., 2015). In the VRP case, the particles present the vehicle route. In which a particle encodes information of the most preferred customers and the vehicle orientation (Ai and Kachitvichyanukul, 2009).

Local search
Local search algorithms search for better solution by applying a single search path to improve solutions. This means a local search evaluates a number of changes to the solution and applies the most suitable change to become the current solution. The three main methods used for VRP are simulated annealing, tabu search, and (adaptive) large / variable neighbourhood search (Gendreau et al., 2008; Tan and Yeh, 2021). All local search algorithms iteratively search the solution space to find better solutions. Simulated Annealing (SA) finds its nature in matter physics. In metallurgy, solids are heated to a point where all molecules are arranged randomly and cooled until the molecules are "frozen". At the start of the algorithm, all evaluated solutions (better or worse) are accepted. During the algorithm, the temperature is lowered, which causes a lower acceptance rate of a worse solution to limit the probability of selection a new solution to "freeze" the solution at an optimum. For finding neighbouring solutions, improvement heuristics are used, as discussed in Section 3.2.1. Tabu Search (TS) is a metaheuristics which uses a list to determine which solutions are already visited and therefore a tabu to visit in the neighbourhood of the current solution. This is to prevent getting caught in a bad local optima. Similar to SA, TS uses improvement heuristic to determine neighbouring solutions. Variable Neighbourhood Search (VNS) is a meta-heuristic which moves to a different (typically larger) neighbourhood when for the current neighbourhood the optimal solution in the current neighbourhood is found. The Adaptive Large Neighbourhood Search (ALNS) works with structurally different neighbourhoods defined by the corresponding heuristics (Pisinger and Ropke, 2010). This is visualised in Figure 16.

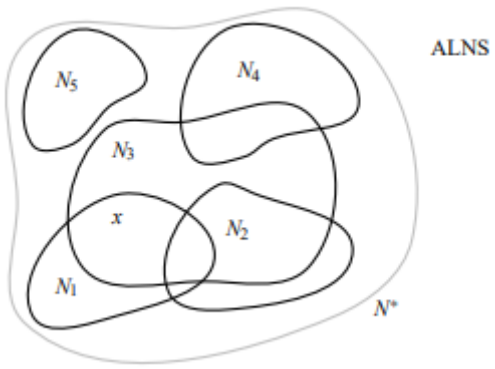


Figure 16: Illustration of the neighbourhoods used by ALNS (Pisinger and Ropke, 2010).

3.3 Decision support

For the planning personnel to make use of the solution design, it is important to investigate theory regarding applications in practice of decision support (in the vehicle routing context). This will be done in Section 3.3.1. Subsequently, we discuss more general how to present information visually in Section 3.3.2.

3.3.1 Transport scheduling

For creating a decision support system, an architecture has to be created. This is the basis for most of the architectures found on this topic. From sources on this topic (Tarantilis and Kiranoudis, 2002; Lacomme

et al., 2021; Santos et al., 2011; Abbatecola et al., 2016), we concluded a decision support system consist of the following components:

- A geographical information handler
- A database
- An arc creator
- A solver
- A User Interface (UI)

All these components are necessary parts of a functioning decision support system. Therefore, we will elaborate on the functionality of these components and how they work within the system in the remainder of this section, starting with the geographical information handler component.

Geographical Information Handler
A geographical information handler works with geographic data and images to present routing outputs on the UI. Tarantilis and Kiranoudis (2002) works with ESRI ArcView GIS, while Santos et al. (2011) uses Google Maps for this purpose. In Lacomme et al. (2021) it includes a geocoder to clean input data and to convert it towards the desired format to use in the decision support system.

Database
The backbone of the decision support system is the database in which data is stored. It contains historical data as well as real-time data (Abbatecola et al., 2016). The historical data is in Tarantilis and Kiranoudis (2002) more regional data, which contains information about vehicles, depots, customer addresses. Lacomme et al. (2021) adds to this information about route plans.

Arc Creator
A third component is there to determine how the route between a and b is travelled. For all decision support systems, distances and travel times have to be determined from a to b in order to optimise the routing. Whether it is called Network Analysis Tool (NAT) (Tarantilis and Kiranoudis, 2002) or Router (Lacomme et al., 2021), it fulfils the same role. The complexity and what happens with this information can also differ. Abbatecola et al. (2016) stores this information as historical data in their database to improve future support information for their decision makers.

Solver
The solver in these systems is the part which constructs solutions based on the input from the other systems. Section 3.2 discusses already the most popular methods for solving VRPs. Table 9 presents the solving methods used in the sources cited in this section.

Table 9: Examples of solving methods used in decision support systems

Source	**Solving Method**
Tarantilis and Kiranoudis (2002)	BATA heuristic
Lacomme et al. (2021)	Compares multiple different solvers
Santos et al. (2011)	Improved path-scanning heuristic & ACO
Abbatecola et al. (2016)	A two-phase heuristic algorithm based on a clustering strategy and a farthest insertion heuristic.

UI
The last part of a decision support is the User Interface. This component consists of the interface where the user can interact with the system. For example, to change the number of vehicles or the order quantity of a customer. The other use is to graphically output the solution from the solver Abbatecola et al. (2016). This together with some general performance metrics of the whole schedule. A third option for interactive decision support is to update the vehicle location repeatedly in the system to allow decision makers to adjust routes while performing.

3.3.2 Visualization

The UI design of the decision support has three parts: The input of the solver, the route output of the solver and, thirdly, some performance metrics of the created output. The design of the two output parts is most important, since planners must be able to interpret the output correctly. Therefore, in the remainder of this section, we focus on the route visualization of routes and the performance metrics output.

Route Visualization
Route visualization has to be appropriate on two levels: High-overview with all routes and stops within those routes. This is how Abbatecola et al. (2016) displays their solution. However, this gives not much detail on individual stops in a route. Karnick et al. (2010) presents a way to provide local context of stops convenient by detail lenses. This problem is similar to the automatic label placement problem, which is a complex problem solved mostly by heuristics. Providing all stop information together also presents the problem of highlighting too much information, which has diminishing returns (Few, 2006, Chapter 3). Therefore, it is more logically to provide this information only when desired, for example, when hovering over a stop.

Performance metrics
Showing performance metrics from the entire schedule gives UI a dashboard element and fits the definition of a dashboard by Few (2004). Sarikaya et al. (2019) provides a recent survey on dashboards. The metrics desired on this dashboard are operational and fits the description of "Dashboards for Decision-Making" from Sarikaya et al. (2019). For every piece of information, it is important to find the best display medium (Few, 2006, Chapter 6). Few (2006, Chapter 7) accentuates the importance of testing for usability, since you will never get everything right in the first try. Therefore, when creating the performance metrics and finding their position in the UI a lot of testing is necessary.

3.4 Conclusion

In this section, we searched for a theoretical routing problem best describing the situation at Company X. Most sources do only describe a part of the problem context, therefore we defined a new routing problem, the Multi-Trip Multi-Compartment Pickup and Delivery Problem Time Windows (MTMCPDPTW). This formulation deals with all essential restrictions derived from Chapter 2. With this formulation are able to translate a practical problem context into a theoretical routing problem. The MTMCPDPTW can eventually be extended with other components if desired.

To solve the routing problem, different solutions methods are evaluated. We investigated heuristics and meta-heuristics which are applied in the VRP context. From this analysis can be concluded that there is a large variety of methods applicable to the problem of Company X. For creating fast solutions, some 2-phase algorithms seem not suitable for the problem instance, because there is more load to be delivered than there is capacity in the fleet. So, there is need for some prioritization, which is best done by a savings algorithm. The last option in this category would be to combine constructive heuristics with improvement heuristics. However, due to the complexity of the problem, improvement operators like move, swaps or k-opts require difficult, thus time-consuming validation checks on feasibility making them inferior over a savings algorithm.

For creating solutions in larger time frame, a meta heuristic is the logical option, when this problem cannot be solved exact in reasonable time. From the large possibilities of meta-heuristics out there, the local search heuristics seem more suitable for this research. Population search algorithms are more difficult to create and thus require more time to build, especially without experience on the particular algorithm. Therefore, local search algorithms seems more appropriate for this research. For this research, a form of neighbourhood search seems the most appropriate and is more often used in recent literature (Tan and Yeh, 2021). Within the group of neighbourhood search solving methods, the ALNS seems most suitable for the problem context. The adaptive selection of destroy and repair heuristics is especially beneficial for a complex problem context, because it is difficult to select the most advantageous heuristics upfront and throughout the procedure maybe a different selection of heuristics is preferable.

As described at the beginning of the section, there is no theoretical formulation of similar to the problem context of Company X in literature. In general, literature of complex VRPs or PDPs is limited. On top

of that, literature on implementation of routing problems into decision support systems is limited as well. Therefore, it is difficult to gain many insights in how researches cope with practical problems in vehicle routing. An insight which can be retrieved is that complex problem contexts are rare, or that researches investigate complex contexts only in simplified form. The application of routing problems into practical decision support systems is within the solver component of decision support system.

The solution method has to be implemented into a decision support system to solve the problem identified. Literature shows that a decision support system consist of 5 components. For design of the UI, we concluded that we present route and/or stop information only when desired, and that presenting the performance metrics in the UI requires testing.

4 Solution Design

This chapter presents the design of the solution for the routing problem at Company X. Therefore, we start with introducing the data-set used to build this solution in Section 4.1. Afterwards, in Section 4.2 we give the full theoretical formulation of the problem to solve and why this formulation is not used to solve the problem of Company X. In Sections 4.3 and 4.4, we present the solving techniques we created to generated planning solutions for Company X. Section 4.3 presents a constructive algorithm to quickly find a solution to the problem formulation, and Section 4.4 explains an ALNS to search for even better solutions to the problem. Lastly, in Section 4.5 the assumptions and input parameter decision and summarized and explained.

4.1 Data Preparation

In Section 2.3 we described how data is handled within Company X. The data-set used to develop the implementation is retrieved from their ERP. Details of the data-set are given in Section 4.1.1. Afterwards, we applied a method to improve the data structure and with that reduce the number of locations, significantly. This is explained in Section 4.1.2. Lastly, we state the assumptions and limitations of the data in the development of the model.

4.1.1 Data-set

The data-set is in essence the order data of a representative average day for Company X. We only use the orders planned for the own fleet. The data-set contains more than 3.000 distribution orders from different customers. However, most of these orders can be aggregated since those are ordered by the same customer. There are 2 different attribute groups of interest in the data, the first one being order quantity information. This is either expressed in colli or pallets, together with the weight of the order. On top of that, the temperature range requirements of the order is given. The second attribute group of interest is location information. This contains information about city, street, postcode and house number per order. This last group of data required data-cleaning, because multiple different street notations are viable in Belgium. Since the data is clustered based on executed orders, executed orders have to be matched to their planned route in order to determine the planned transport quantity accurately. For Belgian locations, this process is more difficult than for Dutch locations, since the combination of postcode and house number is not unique. This is overcome by matching these orders with the standard route for the already known locations from the system. The remainder of orders matched based on their exact match on postcode, HouseNr and the Jaro-distance similarity score between the street and city names. Jaro-distance is a value between 0 and 1 which represents the similarity of two strings (Jaro, 1989). We assume a threshold of 0.9 to be sufficient to match city and street names with each other. The aggregation of 3.000 orders leads to a data-set with 325 different addresses to be visited by Company X on this day, with a subset of the fleet of Company X.

4.1.2 Location Aggregation

The problem size of 325 locations to visit can be further reduced by aggregating locations of the different customers. This is possible since the customers of Company X are mostly businesses, and businesses are often centralized into designated business areas. This is similar for different addresses on hospital campuses. Because these locations are very close to each other, they are scheduled already in practice in the same route. In this research, we combine the locations on postcode. However, we do not neglect the fact that there may be multiple stops on an aggregated location. Therefore, we apply the following formula to calculate the new service time s (in minutes) of the aggregated location:

$$s_{ag} = \sum_{i}^{n} s_i + 5(n-1)$$

In this formula, s_{ag} is the new service time of the aggregated location; n is the number of customers aggregated; s_i is the service time of customer i and 5 is the fixed time (in minutes) to change the unloading address and start new service again. These 5 minutes transfer time is an estimate, approved by Company X. For example, when 3 locations are aggregated which have 10 minutes service time each, the aggregated location will receive a service time of 40 minutes. The result is that 325 stops can be aggregated to 266 stops, which reduces the problem size of the model formulation (see Section 4.2) significantly.

4.1.3 Distances

One of the parameters presents in all vehicle routing problem is the distance matrix. In this distance matrix are all the distances are given between each pair of locations, including the depot. For 266 different location this results in a necessity to know $266 \times 265 \div 2 = 35245$ distances. On top of that, we also need to know the travel times between locations. This to estimate cost of labour in regular and overtime for drivers as well as to prevent violation of driver work times regulations. This to ensure that for drivers regulations regarding work times are not violated. For creating and testing the model, we initially use estimations for these two parameters. The distance estimation is based on haversine formula with the coordinates of two locations, found in a geocoded Belgium cities coordinates file. The coordinates file is manually extended with coordinates of a few unknown locations from our data. The time spends on travelling this distances is based on a speed of 60 km/h to travel these distances.

4.2 Model Formulation

The model formulated to describe the problem of Company X is given in a 3-index formulation. The basis of the problem is the theoretical MTMCPDPTW as described in Section 3.1.13. However, the formulation is extensively larger due to the additional requirements. This section describes first the essence of the formulation in terms of the decisions possible and the requirements to make them. Appendix A provides the full problem formulation. Section 4.2.2 explains why this formulation is unsuitable for the data-set described in Section 4.1.

4.2.1 Model Formulation essence

In this section, we describe the essence of the model formulation with the objective of the model and the conditions necessary to correctly model the situation. The objective of the model is to minimize the costs of the schedule (see Appendix A.4). The costs of a schedule are build op out of 3 costs factors. The first cost factor is the working time of drivers. This is the sum of working hours per driver multiplied by the hourly cost value of a driver. On top of that, there is a cost factor for working in overtime for a driver. After 6 hours of working time, the hourly costs factor increase in order to prevent overtime working and reducing unequal working times between drivers. The second factor is about the cost of undelivered load. Since, not fulfilling orders is allowed (see Section 2.1.4), an initiative to fulfil orders has to be modelled. This cost of undelivered load is a fixed price per pallet. The sum of both cost factors determines the objective value of the model.

Achieving this values is done by constraining the decision variables of the model (see Appendix A.3). These constraints model different conditions which need to be checked in order for the solution to be valid. The exact reasons for each constraint are found in Appendix A.5. Constraints reduce the number of valid solution, but increase the complexity of the model and therefore the solving time.

4.2.2 Model size complications

The size of the problem was already a concern to not be solvable with exact solution methods. The formulation as presented contains $\approx$ 612 thousand parameters, $\approx$ 10.5 million decision variables and more than 41 million constraints. This is for a problem with 266 locations and 36 vehicles. We tested the performance of this formulation with small instances, to check the possibility if solving exact is a viable continuation of this research. This instance contains the orders of a vehicle as planned by Company X. The results are shown in Table 10. The experiments are performed on a laptop with an i7-7th generation processor and 8 GB of RAM with the gurobi optimizer in python. The results show the exponential growth of computation time with the increase of customer and/or vehicles. The 6 customer instances run on in total 43.5 times longer than their 5 customer equivalents. On top of that, an additional (non-trailer) vehicle increases the run time with 107% and 96% for the 5 and 6 customer instances respectively. Therefore, we can conclude that solving this formulation for the complete problem (266 customers 36 vehicles) is not feasible. Thus, other solving methods are required in order to overcome these issues to find sufficient solutions within reasonable times.

The main reason for this large growth is the formulation type in combination with the solving method. The gurobi optimizer uses a branching algorithm. The main idea of branching algorithm is to successively break up the solution space into certain subsets (branches). In order to discard some of these subsets and to reduce the solution space, lower bounds for the objective function (that shall be minimized) over the

Table 10: Experiments Model Formulation Computation Time

No. Customers	**No. Vehicles**	**No. Constraints**	**Trailer?**	**RunTime (s)**	**Nodes**	**Iterations**
3	1	355	No	0	2.115	25.012
4	2	1.036	No	8	21.630	386.733
5	1	725	No	110	451.336	10.236.856
5	1	935	Yes	168	704.527	15.934.685
5	2	1.434	No	228	652.508	13.892.004
6	1	958	No	5.410	20.999.711	438.048.260
6	1	1.088	Yes	5.984	19.503.729	465.575.950
6	2	1.896	No	10.614	26.994.385	498.432.407

subsets are calculated (Theurich et al., 2021). If the lower bound of a subset is larger than an already known objective value of a feasible point (upper bound), then this subset is removed. The general 3-index PDP formulation suffers from symmetry and a weak linear relaxation.

Symmetry is present in this problem since most of the vehicles are similar. Consequently, it is hard for the branching algorithm to find out whether it is optimal to assign orders to these similar vehicles, since these are equally optimal. Therefore, it cannot exclude one option as suboptimal and has to continue evaluating both solutions.

The concept of weak linear relaxation is more complex. A relaxation of the problem is created by a branching algorithm to find a lower bound of the solution (e.g. all routes combined cost at least X). It simplifies binary and integer constraints into continues constraint (for binary continuous between 0-1). In a strong relaxation, this bound is close to the optimal value. However, in the case of this is far from the situation. The relaxation is weak since a lot of constraints are affected by these relaxations. The most explanatory is the decision variable $X_{i,j}^{k}$ which determines if you travel from location i to j with vehicle k (1) or not (0). This variable is very sensitive to the relaxation, since then you can provide it with a value 0.5 in a relaxation. Since, this formulation has a lot of these variables and therefore becomes a weak formulation.

On top of the symmetry and weak linear relaxation suffering, the number of variables can become considerably large as the numbers of requests and vehicles increase (Furtado et al., 2017). Both these factors have severe impact on our formulation as shown in by the experiments. This could be overcome by changing from a 3-index formulation to a set partitioning formulation (see Section X) with column generation techniques. However, these formulations are difficult to make and even more difficult to optimize. Therefore, this technique is not chosen to solve the problem of Company X. Thus, (meta-) heuristics are required in order to overcome these issues to find sufficient solutions within reasonable times.

4.3 Constructive Algorithm

The constructive algorithm is a procedure to construct a solution. This procedure consist out of 5 steps which are shown in Figure 17. In this section, we discuss the logic of each step. Starting with the logic of Standard route construction in Section 4.3.1 and how these orders are positioned in Section 4.3.2. Section 4.3.3 discusses the logic to select which small orders to deliver. Lastly, Section 4.3.4 describes how remaining orders are added to vehicles. A technical description of the procedure can be found in Appendix B.

4.3.1 Standard Route Construction

The standard route construction is the process of assigning the standard routes to the actual routes. Firstly two lists are created containing either the orders with a standard route, or which does not have a standard route, called the unplanned order list. An order has a standard route if it has a vehicle assigned to it. It can also have a position assigned to it. When creating the standard route, this position has to be respected by the procedure. For example, in the case of Company X, orders placed in the trailer are will have position 1 in the standard route, since the trailer has to be detached before the rest of the vehicle

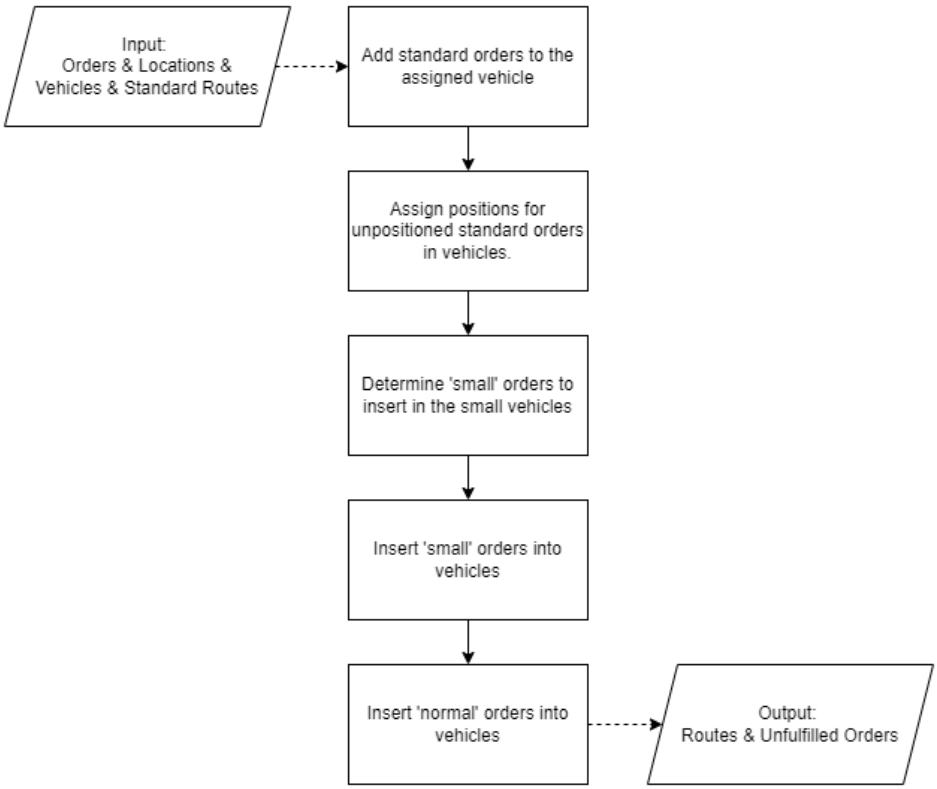


Figure 17: Visualization how the Constructive Algorithm produces a schedule

load can be accessed. However, most orders of Company X do only have a standard vehicle and not a standard position in the vehicle and therefore we have to find positions of the unpositioned orders.

4.3.2 Position assignment

For determining the position for each orders, we evaluate all possible configuration given the orders in the vehicle. If a standard route has 5 unpositioned orders, we evaluate all 5 factorial (120 unique routes). If one of those was positioned, we would only evaluate 4 factorial (24 unique routes). We evaluate the routes on distance between locations. For each unique route, we calculate the total travel distance. We pick the configuration of the vehicle based on lowest total travel distance.

4.3.3 Small Order Selection

The second step is determining which small orders to deliver and which small orders to not consider for insertion into the route of the small vehicles. This pre-processing step before inserting orders into vehicles is beneficial in two ways, computation time and outcome control. It reduces computation time because it limits the number of orders which have to be considered for insertion, which is more computational heavy (see Section 4.3.4). It also gives more control on which orders are going to be planned in the small vehicles and which not. This is desirable since standard routes of small vehicles are often routes within the larger cities (Brussels, Antwerp etc.) and inserting orders outside these cities is highly undesirable because of traffic considerations.

To select the orders which are inserted into the small vehicles, the distance of the order to each small vehicle is calculated. The coordinate point of the vehicle is calculated by taking the weighted average coordinate of the orders present in the standard route of the vehicle. The weight is determined by the load of the orders. This results in a matrix of distances between order and vehicle centre points. For each order, the lowest distance is listed. Based on this list, we reduce the list of orders to plan for small vehicles until we can fit the orders in the vehicles. So the orders most distant orders are not scheduled.

4.3.4 Order Insertion

Before we insert orders into vehicles, we determine the order to insert. This is the largest in terms of pallets on the list. Then we calculate the cost for each of the vehicles, the costs of adding this order to feasible vehicles. A vehicle is feasible if it still has capacity to fit this order and that it supports at least half of the order load. When determining the costs of inserting an order into a vehicle, all possible positions within the route have to be considered. The order will be inserted into the vehicle, which increase the costs the least. This process is repeated until all orders are scheduled.

The non-small orders still are often larger than the capacity available. This results in that some of these orders cannot be planned into any route. These are typically the orders with only a few pallets, since those are the lowest on the list of unplanned orders after sorting on load-size. These orders are transferred from the unplanned list onto the unfulfilled list, so we have all unfulfilled order request on one list.

4.4 Improvement Algorithm

The Improvement Algorithm is the meta-heuristic ALNS that searches neighbouring solutions adaptively. This process and the characteristics of this particular ALNS are discussed in Section 4.4.1. Subsequently, Sections 4.4.2-4.4.5 discusses the initial procedures used to create new solutions from an existing solution.

4.4.1 ALNS Procedure & characteristics

The Adaptive Large Neighbourhood Search (ALNS) has a special position in the literature. When first introduced by Ropke and Pisinger in 2006 it was categorized as a meta-heuristic. Today ALNS is considered a hyper-heuristic because it not only guides and modifies the operations of subordinate heuristics, but also to select and/or generate heuristics. Figure 18 shows the general procedure of the ALNS adapted towards the specifics of Company X. A technical description of the algorithm can be found in Appendix C together with the technical descriptions of the destroy and repair methods (see Section 4.4.2-4.4.5). In the remainder of this section, we cover four of the characteristics of a ALNS. Those characteristics are marked numbered 1-4 in Figure 18 to the corresponding action or decision.

Stopping and Updating Criteria (1)
The stopping criterion is the value which determines the end of the algorithm. The stopping criterion can be based on either performance or on (run)time. A performance based stopping criterion is, for example, stopping after x unsuccessful iterations. While a time based stopping criterion is for example stopping after x iterations. In this research, we choose the stopping criterion based directly on run time, so the algorithm stop iterating after x seconds have passed. This is done to have maximal control on the run time, which is an important input factor for Company X. Consequently, the number of iterations is more variable depending on the chosen heuristics and their corresponding run times. We decide to update the update criterion not on a fixed number of time but on a fixed number of iterations. This is because if we have longer run time heuristics for an iteration, the number of iterations in the update time can be undesirably low, leading to very unstable probabilities because of some fortunate iterations. To find good values for this updating criterion, experimentation is necessary and discussed in Section 5.3.

Degree of destruction (2)
The degree of destruction is an important parameter for any neighbourhood search, including the ALNS. It is the number of variables removed from the current solution by the destroy heuristic. In the case of a VRP, this is a stop within a route of a vehicle. The degree of destruction can range from 1, only removing 1 stop in only 1 route. All the way to the problem size, which means that the whole solution is destroyed and that the repair heuristic has to construct a complete new solution. When the value is chosen too small, it cannot search the search a large neighbourhood, which cause worse performance. When the neighbourhood is too large, it results in too much poor-quality solutions, which consume time which cannot be used to intensify the search in a promising neighbourhood. Therefore, some experimentation will be done on this parameter in Section 5.4.

Roulette wheel (3)
In ALNS the roulette wheel is the guidance mechanism which determine the selection of heuristic to destroy and repair the new solution in each iteration. The roulette wheel has 5 parameters to determine the choice of heuristics on the past behaviour of the specific heuristic. The first parameter is the weight.

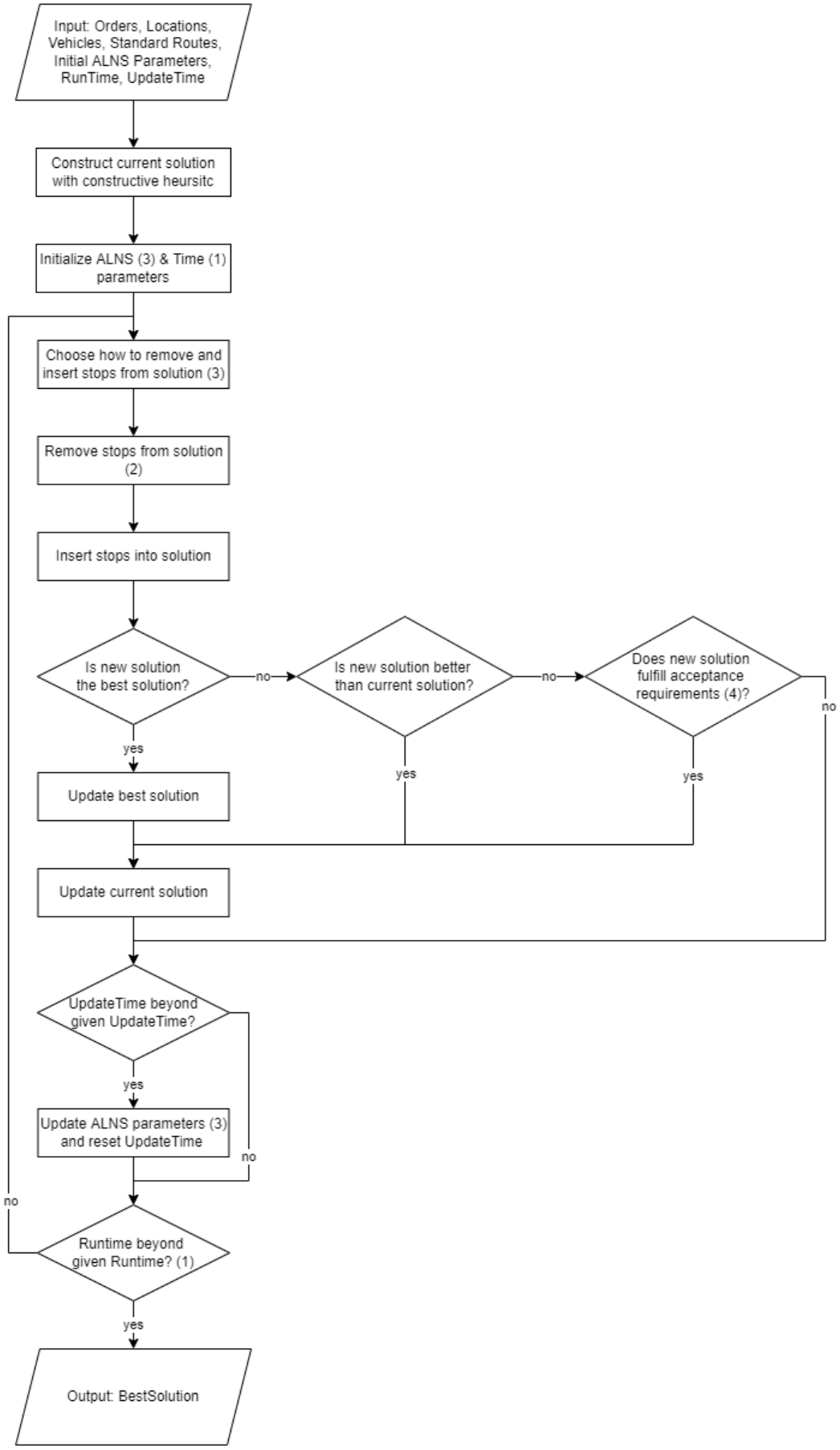


Figure 18: Visualization how the Improvement Algorithm produces a schedule

The weight is a measure of success in past iterations and is updated after a fixed number of iterations with the following formula for each heuristic i:

$$W_i = W_i(1 - \rho_i) \; + \; \rho_i \frac{\pi_i}{\omega_i}$$

The formula is a weighted average on the previous and current success of the heuristic weight by the roulette wheel parameter ρ. The larger ρ, the more important, current success of the heuristic is for determining the weight and vice versa. With this newly calculated W_i, the probability of selection p_i can be calculated. The selection probability is the share of W_i within the total weight of heuristic in the category.

The success rate π_i is updated after each iteration and is dependent on the quality of the solution. Because of the acceptance method we use, there are 4 options to increase the success factor for heuristic i, which are displayed in Table 11. The first options are more successful and are therefore rewarded with a larger increase of success rate. Section 5.5 discusses the exact determination of these values. The last parameter is the usage rate ω_i of heuristic i. This value is simply increased after each iteration and reset together with π when the weights and probabilities are updated.

Table 11: Option overview for updating the success factor

Description	**Update formula**
New Solution is global best	$\pi_i = \pi_i + \sigma_1$
New Solution is current best	$\pi_i = \pi_i + \sigma_2$
New Solution is worse, but accepted	$\pi_i = \pi_i + \sigma_3$
New Solution is worse, and rejected	$\pi_i = \pi_i$

Acceptance method (4)
The acceptance method or acceptance strategy determines whether a solution is accepted as the current solution despite the fact that the solution value is worse than that of the current solution. This factor influences the diversification process of the ALNS. The literature provides many possible acceptance strategies (Santini et al., 2018). In this research, we choose on the most intuitive of the best performing methods, the threshold acceptance. If the acceptance value is lower than the threshold, then we accept the solution, otherwise we reject. The acceptance value is determined by the normalized gap between the new and the best solution and the threshold decrease linear from 1 to 0 in the algorithm. This results in that worse solutions are less likely to be accepted as time progresses.

4.4.2 Random Destruction

Random destruction is the first destroy heuristic and removes stops from the solution at random. Firstly, a vehicle is chosen and then afterwards a stop within the vehicle. This process is repeated until the degree of destruction is reached. Not all stops are viable to destroy, the stops of orders which are standard loaded in trailer are bound to the vehicle and therefore not be destroyed. When selecting a trailer vehicle, additional checks happen to ensure these orders are not removed from the vehicle.

When removing an order from an existing solution, additional checks and updates have to be done to ensure that the solution is still feasible. This holds for all destruction heuristics. For example, the vehicle configuration has to be reconfigured given the pallets remaining in the vehicle. The same holds for the cool unit if the removed order has load in there. What also may happen is that there is no room available for pallets of remaining orders which previously could not fit. These pallets have to be added to create sensible destroyed solutions. Other variables which have to be adjusted are the arrival/departure times and travel distances of remaining orders within the vehicle, as well as work and drive times of the vehicle.

4.4.3 Greedy Removal

Greedy removal is more complex heuristic than random destruction. The greedy removal heuristic removes the least profitable stops from the solution. First, the costs of each stop has to be determined. This is the costs of not transporting the load minus the savings achieved by not travelling to the location of the stop. This is done for every stop, except the orders which are loaded into trailers. The stop with the highest costs is removed and the cost list is updated on the new solution. This process is repeated until the degree of destruction is reached.

4.4.4 Greedy Repair

Greedy repair inserts the most profitable order into the vehicle. We base this profitability on the additional pallets delivered, minus the additional costs of the new stop. These additional costs are calculated for all unfulfilled orders into all vehicle. Based on this matrix, we insert the order with the lowest additional costs into the route and update the additional cost matrix. Then, we update the additional costs on the new solution. This process is repeated until the costs of the order with the lowest additional costs is larger than 0, e.g. the costs will increase rather than decrease.

To check the costs of insertion, we first have to account for the current configuration and load for the vehicle, as well as for the possible insertion position and travel times. Sometimes the vehicle load cannot be added fully when the capacity is available. This can occur because of the compartment restrictions. Also, it can happen that the time windows cannot be satisfied and that makes the solution infeasible.

4.4.5 Regret Repair

Regret repair works similar very similar to greedy repair, except how the order to insert is chosen. From the additional costs matrix, regret values for each order are determined based on the following formula:

$$regret = \sum_{k \in K^-} (costs_k - costs_i)$$

In this formula, i is the vehicle with the lowest additional costs and its costs are expressed in $costs_i$. The set K^- is the set of vehicles without the vehicle with the lowest costs (i). The regret larger the regret the more urge there is to insert this order into the cheapest vehicle right now, since there are not many (or very worse) alternative vehicles for this order to insert into. The order with the largest regret and a feasible vehicle to insert into, is chosen to be inserted into the problem on the location with the least additional costs. Then we update the additional costs on the new solution. This process is repeated until the costs of the order with the lowest additional costs is larger than 0, e.g. the costs will increase rather than decrease.

4.5 Assumptions & Input Parameter Decisions

In this section, we summarize the most important assumption and decisions on input parameters of our solution design. This is beneficial for other researchers to find out what factors are not taken into account in the solution design, so they can eventually elaborate on these characteristics. The summary is also essential for Company X to understand what is taken into account and what not. This section bundles the assumptions and design decisions, which are often already mentioned in throughout the report, into one place. We start with the assumptions in Section 4.5.1 and end with the input parameter decisions in Section 4.5.2. The items are categorized and in Appendix D elaborated explanations of some items are provided.

4.5.1 Assumptions

Location data

1 Delivery locations are aggregated per postcode

2 Location coordinates are generated per postcode instead of the unique address

3 Distances between locations are calculated with the haversine formula (See Section 4.1.3)

4 Travel times are estimated by driving the travel distance at constant speed independent of the vehicle type

Vehicles

5 The travel costs per kilometre are constant and independent of the vehicle type

Driver restrictions

6 Driver costs are fixed for both regular time and overtime

Orders

7 The costs of not serving are fixed per pallet not served

8 Orders cannot be split into multiple vehicles

9 Orders are allowed to be partially delivered

10 Orders are served in constant time indepedent of the order size (see Section 4.1.2)

Problem simplifications

11 Direct orders are not executed before distribution orders

12 Collection orders are not executed before direct orders

13 Insertion costs will not be calculated if an order cannot be fit for more than 25% in a vehicle*

Constructive algorithm

14 Standard orders without position are ordered, based on shortest distance of total route (see Section 4.3.2)

15 For filling the vehicles with "small" not standard orders, close by orders are preferred

16 Largest (total load) non-standard orders will be scheduled first.

Improvement algorithm

17 Standard orders are not fixed to the vehicle

18 Standard trailer orders always remain in their initial vehicle

19 Infeasible insertions are not allowed*

4.5.2 Input Parameter Decisions

Location data

20 "Small" locations can only be served by "small" vehicles

21 "Normal" locations can be served by "small" vehicles as well

22 The constant speed to determine travel times is 60 km/h

Vehicles

23 Trailer orders have to be served before non-trailer orders

24 Standard pallets can be placed in any regular compartment (except cool unit and trailer)

25 Placing load in a cool unit saves 1 pallet

26 The costs per kilometre are €0.50

Driver restrictions

27 A driver works in overtime after 9 hours

28 Maximum working time of a driver is 15 hours

29 Maximum driving time is 9 hours

30 Driver costs in regular time are €32 per hour

31 Driver costs in overtime are €48 per hour

Orders

32 The costs of not serving a pallet are €200

33 Service time per order is 15 minutes (see Section 4.1.2)

Data preparation

34 50 colli represents 1 pallet for orders from a wholesaler

35 40 colli represents 1 pallet for order from a hospital or pharmacy

36 Remaining colli can be inserted on existing pallets (max 10 per pallet)*

37 Standard colli and collies in the 15-25 temperatures are merged together into 15-25 pallets

38 Orders loaded into the trailer have a StandardPosition of 1 in the standard route of this vehicle

Unconsidered practical factors

39 Traffic jams are not accounted for

40 Time-dependent travel speed is also not considered (see Section 3.1.7)

41 Orders cannot be prioritized*

42 Driver (mandatory) break times are not considered

43 Second trips are not considered in the formulation*

* See Appendix D for further explanations on these items.

4.6 Conclusion

In this section, we designed two solution option for Company X to automatically generate transport schedules. In order to do this on a regular basis, a lot of data is required. Most of this data is already retrievable from the current systems, and will be even more easily in the OOMHPA because of the centralization of data. This data includes vehicle, location and order information. For vehicles, this is quantity and specification of the vehicle types. For locations is this the geographical location and time window. Orders data includes the quantity and type of the load.

For the data not available in the system, we made assumptions and simplifications to generated schedules. The travel distance/time data stands out the most. Currently, this is generated based on the coordinates, therefore including coordinates in necessary. However, on the commercial market there are options to gather more accurate data. Another assumption of data is the service time of orders. Currently, this is a flat value, but this assumption can be made more accurately by planners within the decision support system.

In Section 4.2 we defined a 3-index (departure location, arrival location, vehicle) formulation to solve the problem of Company X exact. This resulted in a solvable model, however only in a limited context. Due to the symmetry and weak linear relaxation, this formulation is not solvable in reasonable time for problems with 6 or more customers. Therefore, we limited ourselves to the other solving methods presented in Chapter 3.

We designed two solving methods for the specific problem context at Company X. The first method is a constructive heuristic to find a solution in seconds. This method uses the concepts of a savings algorithm to determine the best vehicle to insert an order into. The base of each vehicle's route is determined by the standard orders within each vehicle. The second solving method is an ALNS tailored to the problem of Company X. This method requires more input and tuning, but has more options to find better solutions

in the time it is given to solve the problem. To find a new solution, the algorithm chooses adaptively between a random and greedy destroy heuristic and between a greedy and regret repair heuristic.

With the two designed solutions and the acquired data, we can evaluate how the solution designs perform with this data. Before comparing the designed solutions with each other and the manual schedule, we have to define on which metric we want to compare solutions and find the best input parameters values for the improvement algorithm.

5 Experiments & Results

In this chapter, we will discuss how we optimize the results of the improvement algorithm. Firstly, in Section 5.1 we identify which performance measures can be used to compare the results of the experiments. Afterwards, we want to discuss initial settings of the improvement algorithm in Section 5.2. These settings are used to find the ideal stopping and updating criteria in Section 5.3. Thirdly, we use these setting to experiment with the degree of destruction (Section 5.4) and afterwards the success rate (Section 5.5). With all these gathered information, we perform a final grid search over all parameters for the improvement algorithm in 5.6. Followed by Section 5.7 discussing the performance of the individual methods within the improvement algorithm. The penultimate section (5.8) compares the results of the manual planned schedule, constructive algorithm schedule and the improvement algorithm schedule with each other. Lastly, in Section 5.9 discusses the robustness of solutions.

5.1 Performance measurement

In order to compare and evaluate schedules, we have to define performance measures which are intuitive to interpret and comply with the way planners currently created their planning as explained in Section 2.1. Therefore, we assign costs factors to the most important measures of a planning. The factors are determined together with the management of Company X. The difficult factor is to assign a cost for not serving pallets. This value is an assumption based on how many additional costs are allowed for delivering additional pallets. The cost factors and their values are listed below:

- costs of working in regular time, €32 per hour
- costs of working in overtime, €48 per hour
- costs of travelling distance in kilometres, €0.5 per km
- costs of not serving pallets, €200 per pallet

These four costs factors are summed to create the Total Costs of a schedule. This performance measures balances the costs factors as given by the proportions of the costs factors as provided in Section 4.5. Therefore, the Total Cost is used in the ALNS to compare the new solution with the current solution. Consequently, the solution with the lowest Total Costs is the final solution of the ALNS. When Total Costs of two experiments are similar, we can also investigate the cost values of the different components in a schedule. This is especially interesting when a solution serves more pallets but at a higher kilometre/driver costs. However, choosing which option is better is difficult without any practical context of the routes of individual vehicles. Therefore, we also consider another performances measure.

Another way to measure the performance of experiments is with the average Total Costs of all performed iterations. This metric in combination with a solution value trend plot shows whether the algorithm finds good solutions with the current settings, or that the best solution was a more fortunate solution within the whole search. A single good solution may be good for this specific problem and settings, but may not deliver good solutions in other circumstances. Therefore, the average solution performance and development in solution values show a more thoughtful imagine on the performance over the total costs of the best solution.

These two measures are the general performance measures used in the Section 5.3-5.6. For the other experiments, other measures may be more valuable. When those measures are used, they are explained in the respective section.

5.2 Initial Settings

In order to start experimenting with setting of our improvement algorithm, we have to find reasonable initial settings to perform these experiments with. Therefore, we investigate ALNS literature on PDPs to find the most used settings. Appendix E shows the literature sources to create the experiments in this chapter. The experiments are performed on the same laptop as described in Section 4.2.2. The runtime of the experiments is fixed for all experiment to an hour, since Company X desires this runtime for distribution planning solution (See Section 3.2).

From these sources, we found that it is most common to use a fixed number of iterations as update criterion. On average, there are between 50 and 250 updates per run. Due to the feasibility requirement and size of the problem, the run time of one iteration with our system (laptop, i7-7th gen, 8GB RAM) is approximately 30 seconds. Therefore, many iterations will not be reached and thus many iterations is between updates is not reachable. Therefore, we set the initial update criterion to 5. This initial value allows for sufficient updates of the parameters within the runtime, while maintaining a sufficient probability for all heuristics to be used within the update interval. For the degree of destruction, the most common approach is a percentage of customers. We apply the most common approach and therefore set the initial value to 15% of customers. The success rate has two common setups. The first one is $\sigma_1 > \sigma_2 > \sigma_3$, the second one is $\sigma_1 > \sigma_3 > \sigma_2$. We choose the first approach since it is more intuitive to value an improvement more over a worse but accepted solution, and it is more common (4 versus 3) in the literature sources of Appendix E. Since we perform limited iterations between updating, we do not want to assign too large differences to the values, resulting in [10,5,2,0] as starting values of success rates. Because of the limited iterations and the limited number of destroy and repair heuristics, we do not want a large initial roulette wheel parameter. Therefore, the common value in literature of 0.1 is a good starting point. This resulted in the starting solution displayed in Table 12.

Table 12: Starting Solution Input Parameters

Iteration best found solution	**Total Costs**	**No. missed pallets**	**Worktime (h)**
54	€38.403	127	272

5.3 Stopping & Updating Criteria Experiment

With the initial values, we can experiment with the updating criterion, the number of iterations between parameter updates. We have chosen the update criterion values in such a way that it maintains sufficient probability for all heuristics, while also have sufficient updates of the parameters. A large update criteria allows for more balanced check of the performance of the heuristics at the costs of adaptability. With the low update criteria values, there is a possibility that a heuristic is not picked. In that case, the weight remains equal. If the other heuristic is successful, the probability to be picked drops even lower. However, when a heuristic produces worse unaccepted solutions (σ^4, see Section 4.4.1) the weight value decreasing. Consequently, increasing the probability of other heuristics to be picked. The experiment settings and general results are shown in Table 13.

Table 13: Setting value and results of the Updating Criteria Experiment

Update Criterion	**Iteration best found solution**	**Total Costs**	**No. Missed Pallets**	**Worktime (h)**
3	67	€37.333	123	265
5	54	€38.403	127	272
7	54	€38.403	127	272
10	104	€38.050	130	250
15	54	€38.304	127	272
20	32	€37.351	120	272

From the Total Cost performance indicator, we see that two out of the six experiments yield a value below €37.500. The solution values of those two have a maximum relative distance of 0.04%. This difference is too small to state that one setting outperforms the other. Therefore, we look further into how those solutions are build up and created.

Figure 19 and Table 13 show that Total Costs value within the first half of the experiment run is higher with a value of €38.403. This shows that the update criterion 5 does not reach a good solution compared to the 3 and 20 update criterion experiments in a one-hour run. In combination with the fact that the average solution values are also larger, the best update criteria remain 3 or 20, the smallest and largest value tested.
Figure 19 shows that the average solution value of update criterion 20 is higher (≈€200) than the value of update criterion 3. From only this performance measure, we cannot decide whether the smallest value

(3) or the largest value (20) is most suitable for further experiments.

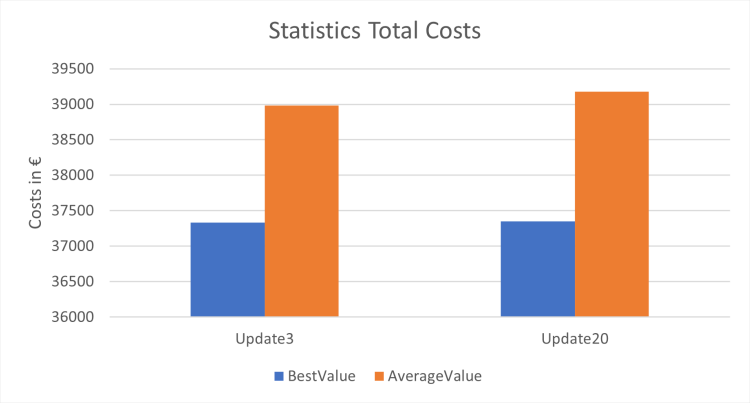
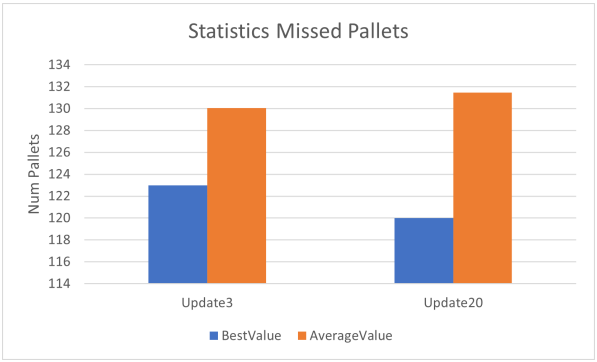
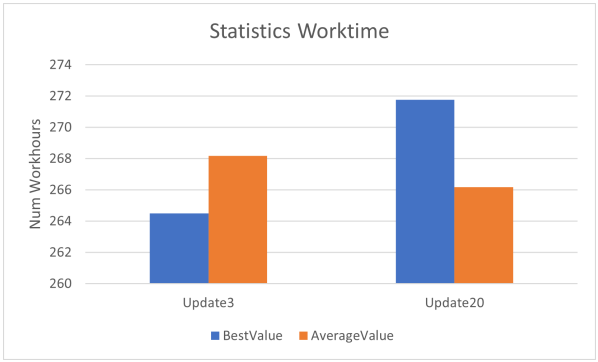


Figure 19: Total Costs statistics of Update Criterion 3 and 20, which have similar best solution values. Only the average solution costs of Update Criterion 20 is slightly higher.

To choose the best criterion between 3 and 20, we have to look more into how the solution is build up. Figure 20 presents the missed pallet and total work time of the lowest cost solution found. The figures show that the best solution with update criterion 3 has more missed pallets (Figure 20a), but compensates this with 7 less work hours (Figure 20b). From the detailed schedule solution, we see that approximately 700 km is travelled for the delivery of those 3 additional pallets. This is the practical trade-off which can only be made by a planner. From a theoretical standpoint, update criterion 3 is more logical. Since a value of 3 leads to ≈ 35 updates per hour of the parameters, which is more in line with the theory found in Section 5.2 than the ≈ 5 updates per hour with an update criterion of 20. In conclusion, we continue with an update criterion of 3 in the Degree of Destruction and Success Rate experiments.



(a) The solution with the lowest costs of Update Criterion 20 finds a solution with less missed pallets compared to Update Criterion 3.



(b) The solution with the lowest costs of Update Criterion 3 finds a solution with work time compared to Update Criterion 20.

Figure 20: Schedule statistic of the best performing criteria in this experiment

5.4 Degree of Destruction Experiment

For the degree of destruction, all searched literature introduced in Section 5.2 use a degree of destruction dependent on the problem size. However, there are some differences in the design. These designs are: flat percentage, random value between two percentages, random value between two percentages with additional customer bounds, flat percentage which decreases during the procedure and lastly, a value between two percentage bound which is dependent on the number of previous rejections. All options are worth investigating. For the bounds however, we do not go below 10% and 50% to reduce the number of experiments. We define the following experiments (see Table 14).

These experiments provide a lot of insights in successful methods and ranges in which the algorithm performs better or worse. In general, we observe that a high degree of destruction performs worse than

Table 14: Experiments Degree of Destruction

Experiment Nr.	**Degree of Destruction**	**Total Costs**	**Average Destruction**	**Minimum Destruction**	**Maximum Destruction**
1	10%	€37.740	17.0	16	19
2	30%	€37.948	50.2	49	56
3	50%	€38.720	85.0	81	94
4	10% decreasing to 1 customer	37.740	6.5	1	19
5	30% decreasing to 1 customer	37.921	15.0	1	56
6	50% decreasing to 1 customer	38.402	20.2	1	94
7	random between 10% and 25%	€38.573	28.7	17	44
8	random between 17.5% and 32.5%	€38.574	42.4	30	58
9	random between 25% and 40%	€38.383	55.5	40	72
10	between 10% and 25%, increased with rejected solutions	€37.740	17.0	16	19
11	between 17.5% and 32.5%, increased with rejected solutions	€38.626	29.4	29	33
12	between 25% and 40%, increased with rejected solutions	€38.113	42.1	40	47
13	random between min(20,10%) and min(60,30%)	€38.360	33.4	16	51
14	random between min(40,20%) and min(90,45%)	€39.101	54.5	32	78

a destruction rate below 30% customers (±50 customers). We think the reason for this behaviour is that the repair heuristic repair per order, which results that larger orders are more often profitable to insert first. Since, these orders have the highest potential costs savings. Consequently, good combinations of small orders are less likely to be inserted together in the vehicle. The experiments with random degree of destruction (7-9,13,14) are the worst performers, even the experiments (7,13) with the lower degree of destruction are outperformed by others with more than €500. With these clear results remain only 4 settings viable to further investigate, which are from experiment 1,2,4,10.

The statistics of these solutions are compared in Figure 21 together with the result of the 15% degree of destruction run in the update experiment. From these results, we can see that experiment 1,4 and 10 show remarkable similar results. Figure 22a shows the solution value development of experiment 1 and 10. From the figure, we can conclude that these experiments follow a similar development throughout 95% of the runtime. Only at the end, the rejected solution component of experiment 10 is used. This is because of the solution acceptance method used (see Section 4.4). Therefore, without changing the acceptance criterion, using a increase of degree of destruction after rejection is insignificant. Subfigure 22b compares the solution development of experiment 1 and 4. From these trajectory curves can be seen that the last iteration face of experiment 4 is not beneficial as exploitation since the solution values hardly change.

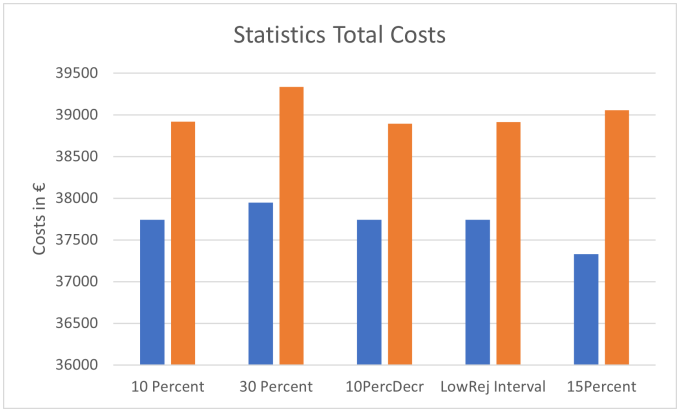
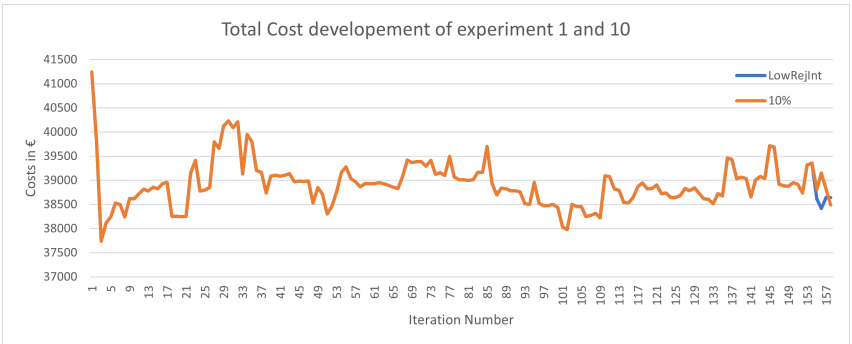
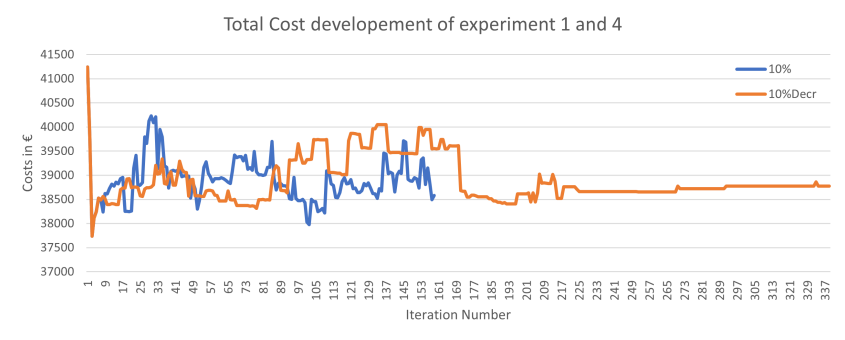


Figure 21: Total Cost statistics of best performing Degree of Destruction experiments and the best and the 15% experiment from the Update Criterion Experiment showing similar results for the 10%,10%Decr and Low rejection interval experiment.

From all the figures, we observe that all 10% runs reached their global best early in the procedure and that those experiments had trouble finding solution under €38.000 again. These runs probably have a too low degree of destruction to successfully find new promising neighbourhoods within the time frame. The statistics figure also shows that the solution found with 15% degree of destruction found a significant lower solution value. Therefore, we conclude that a degree of destruction between 10% and 30% performs best. We do not see good results with (mixed) random degree of destruction intervals. A decreasing



(a) Solution value development curves of the 10% flat experiment (1) and the low rejection interval experiment (10) showing a almost identical total cost development.



(b) Solution value development curves of the 10% flat experiment (1) and the 10% decreasing experiment (4). Experiment 4 shows an ineffective exploitation with low degrees of destruction in the second half of iterations.

Figure 22: Solution value comparison of similar performing experiments (1,4,10)

degree of destruction may work to exploit the solution better, however the minimum degree must be at least 5 customers (2-3%) to be able to make significant impact on the solution. Will we investigate this further in the grid search in Section 5.6, for now on, we will continue experimenting with a flat 15% degree of destruction.

5.5 Success Rate Experiment

Creating experiments to find the success rate is more difficult. In literature (Appendix E) there are namely 2 different setup for the sigma values $\sigma_1 > \sigma_2 > \sigma_3$ and $\sigma_1 > \sigma_3 > \sigma_2$. (Li et al., 2020) is the only paper with dynamic success rates. Since this is not common in literature and difficult to apply, we will not investigate this further. However, the high σ_1 values used in this study might be good to use in our study. Majidi et al. (2019) use σ values between zero and one. This might be a good range of values to use and therefore we will experiment with these low σ values.

The enormous possibilities of different values for all three sigma values cause that we have to focus on the effects of high and low values of σ and difference between σ values rather than the precise values itself. From the sources of Appendix E the values of σ_1 are 1.5 up to 7 times larger than the values of σ_2. The σ_2 values are a 0.5 up to 6 times larger than σ_2 values. With these difference values, we create an experiment design with three starting values of σ_1, three relative differences (high, medium, low) between σ_1 and σ_2 and three relative differences between σ_2 and σ_3 (see Table 15). In total, this setup creates three cubed is 27 experiments. However, we consider three experiments invalid since the high, high setting for the relevant distances leads to $\sigma_3 > \sigma_1$. Therefore, we remove those from the set, resulting in 24 experiments which are shown with their general results in Table 16.

Table 15: Success rate experiment setup table showing the input values to create different experimental settings. Note that a lower relative distance results in higher values of σ_2 & σ_3.

Ordinal Scale	σ_1	**Relative Difference σ_1 and σ_2**	**Relative Difference σ_2 and σ_3**
High	70	1.5	0.5
Medium	20	3	2
Low	1	7	6

This experiment provided a lot of different results and findings with the different inputs. Figure 23 presents the statistics of the different σ_1 values we tested (70,20,1). From this figure, we see that a higher σ performs better in finding the lowest total costs (averaged over all seven experiments with this value). The averaged average total costs are marginally better when $\sigma_1 = 20$.

To investigate which values of σ_2 and σ_3 performs best, we compare the values by averaging the setting of σ_2 and σ_3 for each σ_1 values tested. Resulting in seven averages, because the setting high, high is

Table 16: Success rate experiment settings and their best total cost value

Experiment Nr.	**Success Rate Values**	**Total Costs**
1	[70,47,23]	€37.513
2	[70,47,8]	€37.513
3	[70,23,47]	€37.977
4	[70,23,12]	€37.513
5	[70,23,4]	€36.894
6	[70,10,20]	€37.513
7	[70,10,5]	€37.513
8	[70,10,2]	€37.513
9	[20,13,7]	€37.513
10	[20,13,2]	€38.295
11	[20,7,13]	€37.287
12	[20,7,3]	€37.711
13	[20,7,1]	€37.602
14	[20,3,6]	€37.730
15	[20,3,1]	€37.730
16	[20,3,0.5]	€37.557
17	[1,0.7,0.3]	€37.660
18	[1,0.7,0.1]	€38.432
19	[1,0.3,0.7]	€37.159
20	[1,0.3,0.2]	€37.351
21	[1,0.3,0.05]	€38.500
22	[1,0.15,0.3]	€37.351
23	[1,0.15,0.07]	€38.500
24	[1,0.15,0.02]	€38.489

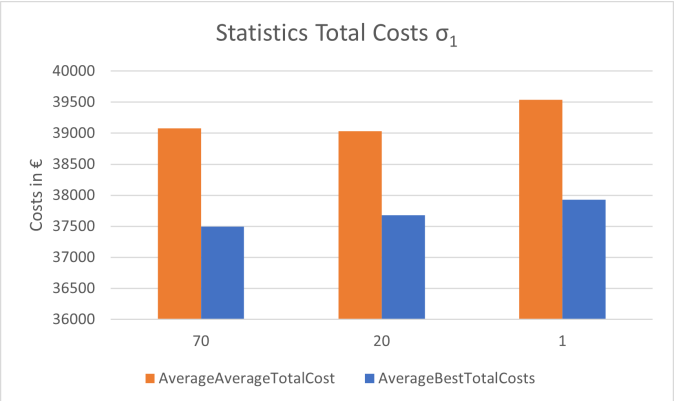
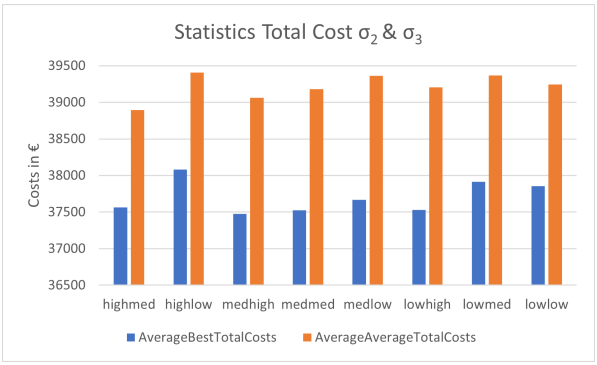
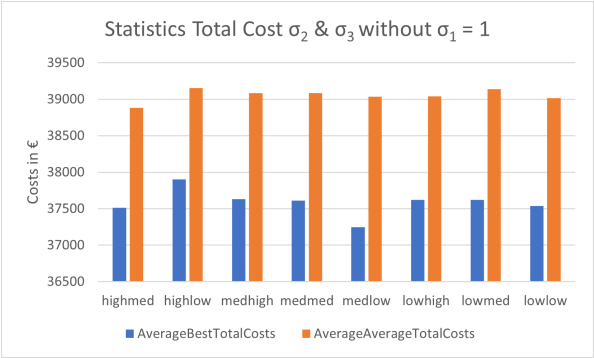


Figure 23: Average Total Cost statistics of the different σ_1 values tested, showing better performance for a higher σ_1 values.



(a) Average Total Cost statistics of the different combinations of σ_2 & σ_3 showing the best results for the medium, medium and medium, low setting.



(b) Average Total Cost statistics of the different combinations of σ_2 & σ_3 tested without the values of $\sigma_1 = 1$ experiments. The medium, low setting clearly finds the lowest best costs values.

Figure 24: Average Total Costs statistics for the different combinations of σ_2 and σ_3

not allowed. Figure 24a shows the cost statistics of those setting. Subfigure 24b shows the average values, where $\sigma_1 = 1$ is excluded from the average calculation because it is outperformed by the larger σ_1 values. In Subfigure 24a, the medium, high and medium, medium experiments show the best costs. In Subfigure 24b, the medium, low setting outperforms the others since the worst performance of experiment 21 increased the average of medium, low in Subfigure 24a substantially.

The best performing experiment setting overall is the high, medium, low setting of σ_1, σ_2 and σ_3 respectively. This setting is the only setting so far finding a best solution value below €37.000. In the Total cost development graph (see Figure 25) we see that this value is only reached at the last few iterations of the procedure. Figure 26 shows the probability development of each of the used heuristics in this experiment. From this figure we observe two stages, the first stage of approximately 50 iterations uses almost exclusively the greedy repair heuristic because of its success in the first few iterations. Only after 50 iterations, the probability of using regret repair is increased step by step towards $\approx 40\%$. The destroy heuristics stay more balanced between 40%-60%. However, at the end, we see a higher probability of random destroy. This is to explore the neighbourhood solution less structured, resulting in finding a neighbourhood of solutions below the €37.000 which no other experiment so far has discovered.

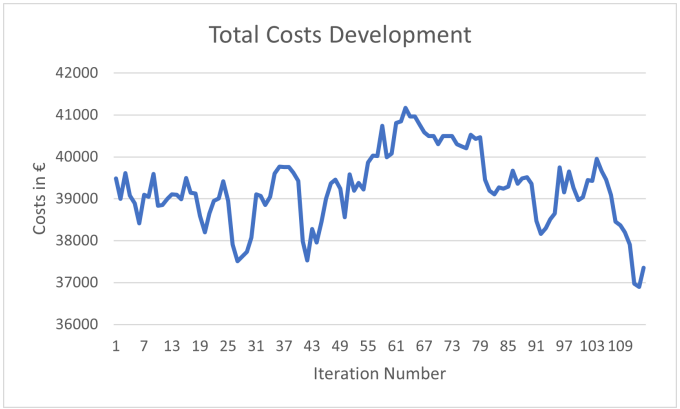


Figure 25: Development of the Total Costs performance measure of the best performing experiment (5)

From all 24 experiments, we observe a high σ_1 in combination with relative large gaps finds the best solutions. A consequence of this behaviour is that good performing heuristics are more likely to be chosen. Concluding that high adaptability on performance results in better solution values. We want to investigate in the grid search whether increasing the gaps between σ_1, σ_2 and σ_3 results in even lower total costs values or if this was observation was a one time success.

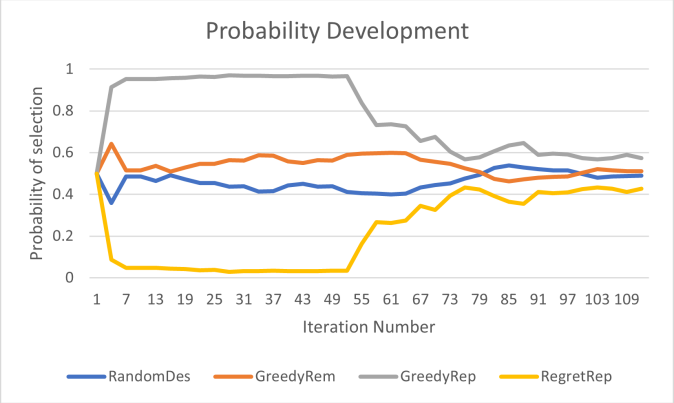


Figure 26: Probability development of the four heuristics in the best performing experiment (5)

5.6 Grid Search

In a grid search, we combine different experiment values with each other to find the best combined settings. This is necessary since parameter settings are not independent of each other. For example, both the roulette wheel parameter and the success rates influence the weight and therefore choosing probability of heuristics. In this experiment, we test two settings in three different categories. This is a relative small grid search with eight experiments. The first category is the degree of destruction. Section 5.4 discusses that a decreasing percentage may provide good results if the lower bound is not lower than 5 customers. We investigate this hypothesis by comparing a flat 20% degree of destruction which is a promising but uninvestigated degree of destruction with a decreasing degree of destruction from 20% to 10%. The second experiment category is overall higher success rate parameters. In Section 5.5 we discussed that high gaps between the different σ values and a high start value is beneficial. Therefore, we compare the original σ values with σ multiplied by 1.25 for all σ. Resulting in the following σ list for σ_1, σ_2 and σ_3: [88,29,5]. The last but uninvestigated input parameter we experiment with is the roulette wheel parameter ρ. As default, we used 0.1, but in this grid search we want to check whether a higher values yields improvements. Therefore, we test a value of 0.3. This results in 2^3 equals 8 experiments which are presented in Table 17. The results are show in Table 18.

Table 17: Input parameters of the grid search

Experiment Nr.	**Degree of Destruction**	**Success Rate Values**	**Roulette wheel parameter**
1	20% Flat	[70,23,4]	0.1
2	20-10% Decreasing	[70,23,4]	0.1
3	20% Flat	[88,29,5]	0.1
4	20-10% Decreasing	[88,29,5]	0.1
5	20% Flat	[70,23,4]	0.3
6	20-10% Decreasing	[70,23,4]	0.3
7	20% Flat	[88,29,5]	0.3
8	20-10% Decreasing	[88,29,5]	0.3

We can derive directly two results from the grid search experiment. The first is that none of the setting beats the best solution found of €36.894 in experiment 5 in the success rate experiments (Section 5.5). The second clear results is that experiment 6 of this grid search find a worse Total Cost value compared to all others. This can be seen in the statistics in Figure 27. However, the average Total Costs of all iteration values is the fourth best. Therefore, we cannot state that the setting is worse, but that the experiment did not find a neighbourhood with solutions below €38.000, whilst others found this neighbourhood. This can be seen in Figure 28, in which the Total Cost values of experiment 8 ones peak and ones finds a larger neighbourhood below €38.000 whilst experiment 6 does remain around solution values of €39.000.

Table 18: Results grid search per experiment

Experiment Nr.	Total Costs
1	€37.844
2	€37.752
3	€37.586
4	€37.822
5	€37.757
6	€38.459
7	€37.757
8	€37.434

Therefore, we consider it more an outlier and will not take this experiment into account when comparing the performance of the different factors tested in the grid search.

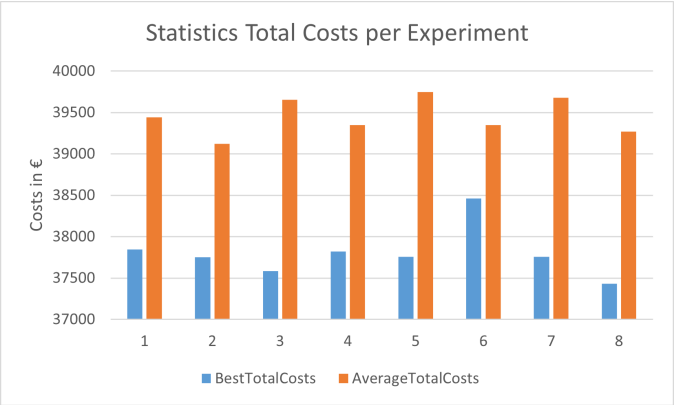


Figure 27: Total Cost statistics of the grid search experiments showing a worse result for experiment 6 for the BestTotalCost value.

We investigated the performance of the different factors tested in the grid search by taking the averages of the experiments with the specific factor. The statistical results are presented in Figure 29. If we compare the different factors we see that the 20%-10% interval outperforms the flat 20%, the higher success rate (SR) values outperform the lower ones marginally, the same hold for the higher roulette wheel parameter (RWP). This is in line with the individual experiments, since experiment 8 with a percentage interval and high success rate values and high roulette wheel parameter yielded the lowest Total Costs value as well.

5.7 Performance Heuristics

The performance of the ALNS is not only dependent on the input parameters. A large proportion is also dependent on the heuristics used to find new solutions. This makes it useful to investigate the effectiveness of the heuristics. This section will compare the performance of the heuristics from the grid search experiments of Section 5.6. Figure 30 displays a comparison in usage rate of the different heuristics. We observe that greedy repair is more used compared to regret repair. This can be explained because greedy repair shows good performance in the first few iterations and therefore gains a lot of probability percentage over regret repair (95+% over 5-%). This remains in general for around 50 iterations before it stabilize between an interval of 40%-60%. The destroy heuristics are more evenly used, with ± 45% for the random and ± 55% for the greedy approach.
However, this does not provide any insights in how the different heuristics perform. The figures within Figure 31 shows the performance of the heuristics to find the 4 different solution types explained in Section 4.4.1. In Subfigure 31a, we see that random destruction finds more global best solution than the greedy removal, despite the fact that is has a lower usage rate. However, the random destruction produces less current best, which is visible in Figure 31b. Random destruction only produces 33% current best solution, whilst greedy removal produces 42% current best solutions. This partly explains the difference

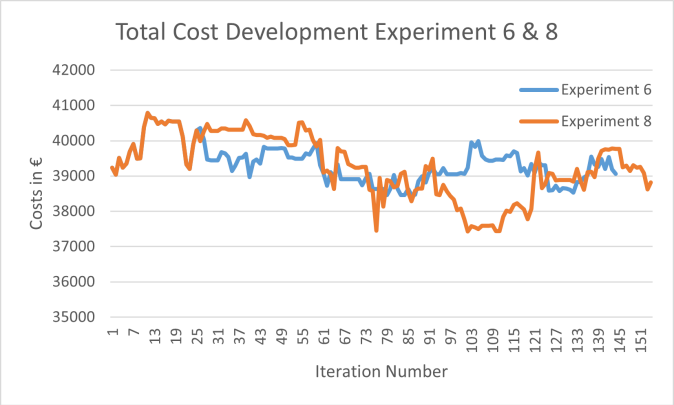


Figure 28: Solution value development curves of experiment 6 and the experiment 8. Experiment 6 finds total costs values around €39.000 whilst Experiment 8 finds values below €38.000.

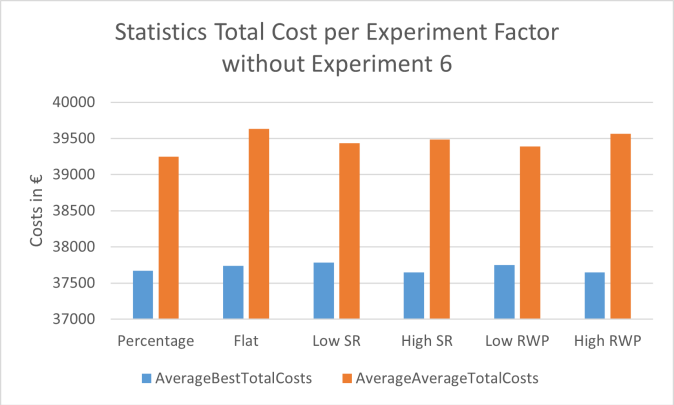


Figure 29: Total Cost statistics of the different factors tested in the grid search. Percentage Interval, High SR and Low RWP outperform their counterpart setting.

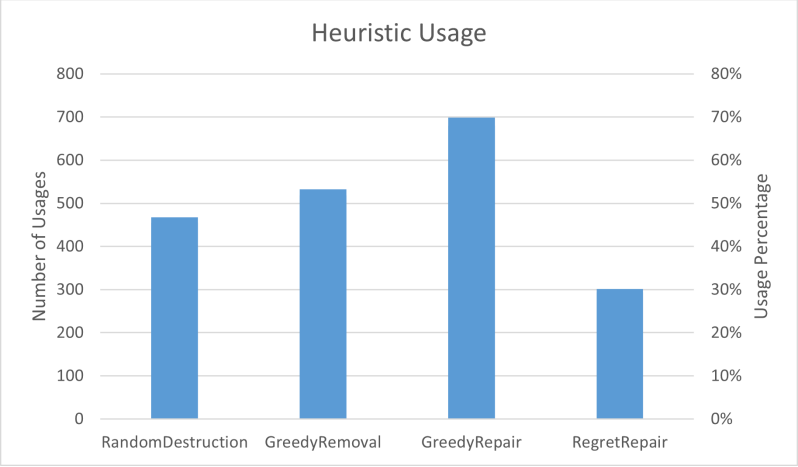
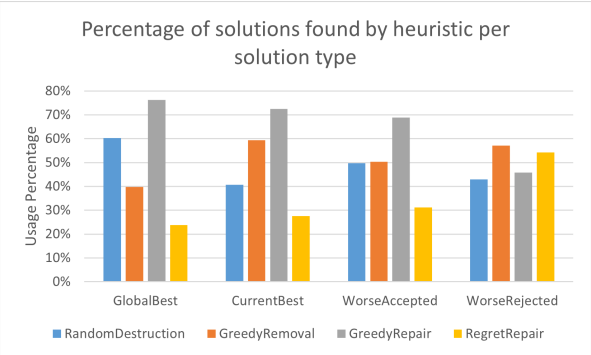
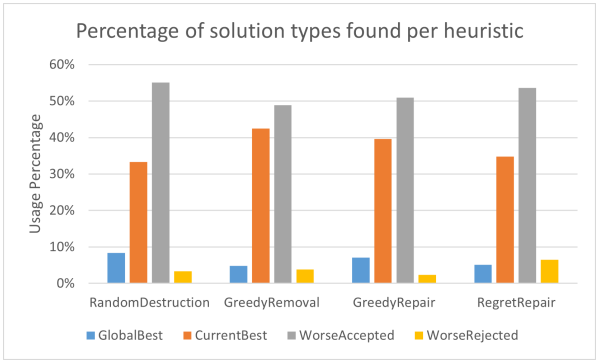


Figure 30: Usage rates of the different heuristics in the grid search showing a high usage rate of the greedy repair compared to the regret repair.



(a) Percentage of solutions found by heuristic grouped per solution type showing a high percentage of global best solutions found by Random Destruction and a high percentage of current best solutions found by greedy removal.



(b) Percentage of solutions type found for each of the heuristics showing that the greedy removal find ≈10% more current best solutions.

Figure 31: Performance comparison of the different heuristics

in why greedy removal is picked more often than random destruction.

The other reason is found by further investigating the iterations in each of the experiments. We found out that a global best solution found from a greedy removal followed faster after a previously global solution, than with a random destruction (see figure 32). This, together with the higher percentage of current best solutions found, explains the preference for the greedy removal over the random destruction. For greedy repair and regret repair, these values are similar and therefore not explaining the difference in preference of greedy repair over regret repair.

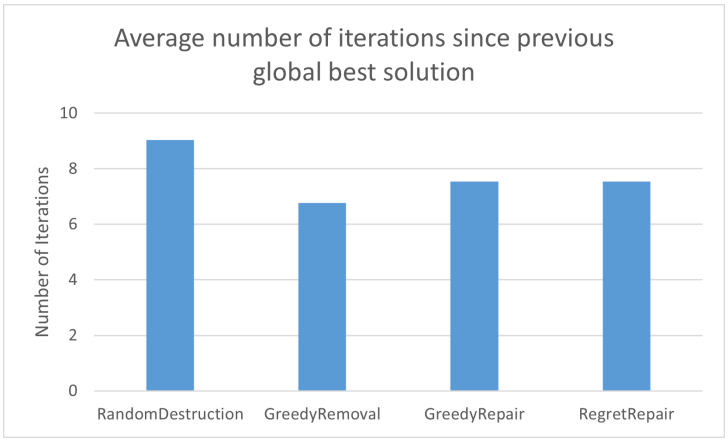


Figure 32: Average number of iterations since the last global best solution when the new global best solution is found by the different heuristics. The greedy removal requires fewer iterations to find a new global best compared to the random removal.

5.8 Schedule Comparison

After tuning the parameters of the ALNS we can compare the schedules created manually on the dataset with the solution created by the constructive heuristic and the solution found by the ALNS. For the solution of the ALNS we use the solution with the lowest Total Costs found so far in this research. Which is experiment 5 of the success rate experiment of Section 5.5 with a Total Costs of €36.894. Because this solution has the lowest Total Costs value, it is most suitable in highlighting the difference in the solutions created by the ALNS compared to the other methods. Firstly, we compare the different performance measures explained in Section 5.1. Lastly, we investigate the effectiveness of standard routes, as suggested in Section 2.1.1.

The most well-defined comparison measure is the Total Costs. Therefore, in Figure 33 we compare the Total Costs of the three schedules. To gain more insights, we split the costs into two categories. The first one is the Unserved Costs, which are the costs of not serving pallets to customers. The second one is called Work Costs, which contains costs for working in regular and overtime, as well as driving the distances. From the figure, we see that the cost savings of the constructive and especially the improvement (ALNS) algorithm is almost entirely by serving more pallets. This is possible in two ways, reducing the empty capacity in the normal load compartments and reducing empty capacity in the cool units. Appendix F presents a large table with all derived information from the schedules.

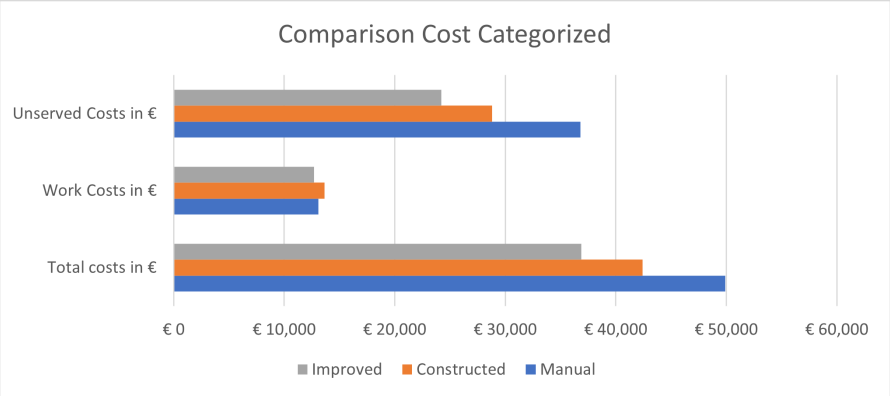
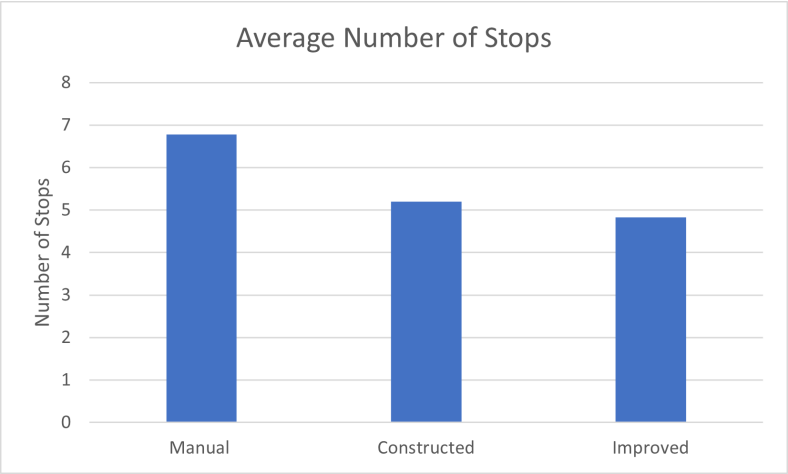


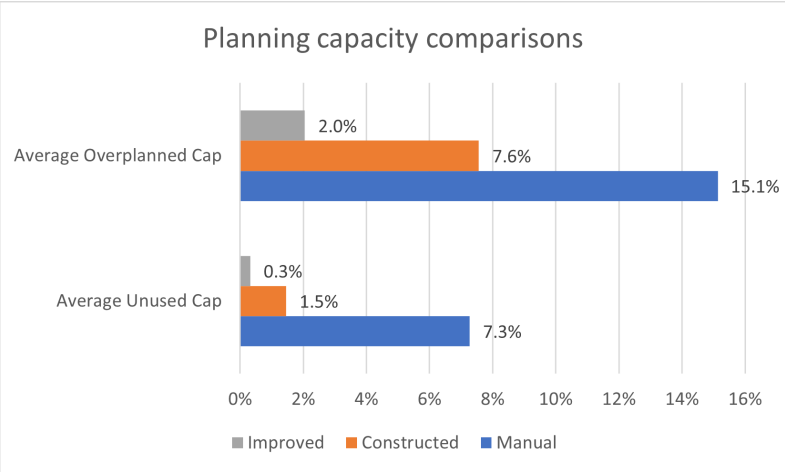
Figure 33: Costs comparison schedules of total costs and a breakdown in two cost components. The costs of unserved pallets cause the difference in total costs for all schedules

When we further investigate the capacity usage, we found that there is not only unused capacity in the schedules. There is also overplanning of capacity, meaning that there are more pallets scheduled on the vehicle than there is capacity. Consequently, there is no unused capacity, but this may lead into unnecessary stops in the routes. A difference in number of stops is visible in Figure 34a, which visualizes that the improvement algorithm solution has on average less stops than the manual constructive schedules. The Overplanned capacity percentage is calculated by taking the sum of unserved pallets of orders in a vehicle and dividing it by the vehicle capacity. Figure 34b shows that the improvement algorithm is the best in balancing the capacity usage without overplanning the vehicles too much. This overplanning occurs mostly in specific vehicles. The box plot Figure 34c highlights this by showing that for the constructive schedule at least a quarter of vehicles has no overplanning and for the improvement solution this is more than half of the vehicles.

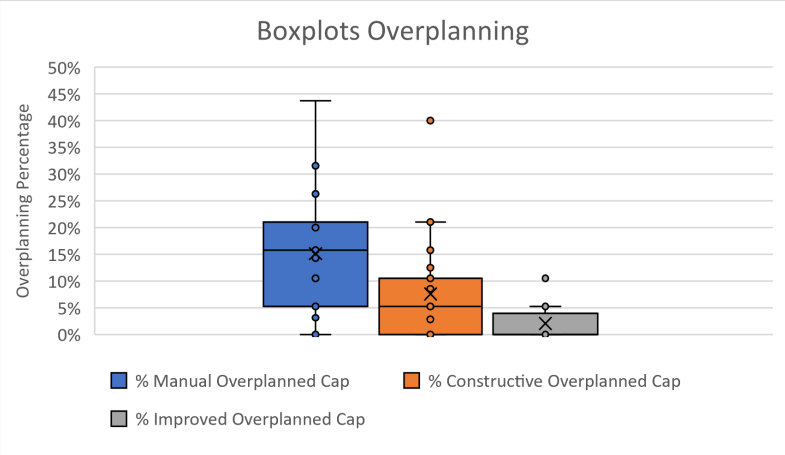
To investigate the effectiveness of standard orders, the first measure to check is the usage of standard orders in the different schedules. The manual and constructive schedules use the most standard orders.



(a) Average number of stops per schedule, showing the highest value for the manual planned schedule.



(b) Schedule comparison of the unused and overplanned capacity averaged per vehicle, showing lower percentages for the constructed and improved schedule.



(c) Boxplot displaying the spread of overplanned capacity percentage per schedule. It shows the reduced spread in values for the constructed and especially the improved schedule. Important to note is that the manual schedule has an outlier of 78%, which is not shown in the graph to maintain general visibility of the figure.

Figure 34: Schedule comparison of the load planned in the routes of vehicles

The constructive schedule contains the most with 125 since in this procedure it is not allowed to move these orders. The manual schedule contains approximately 80% of the standard orders in their standard vehicle. The improvement schedule only contains 17 standard orders, which is around 13% of the total. If we compare this with the costs of the different schedules, we can see that the usage of standard orders can result in improvement in the schedule. However, the schedules with the least Total Costs contains a limited amount of standard orders. These results are visualized in Figure 35.

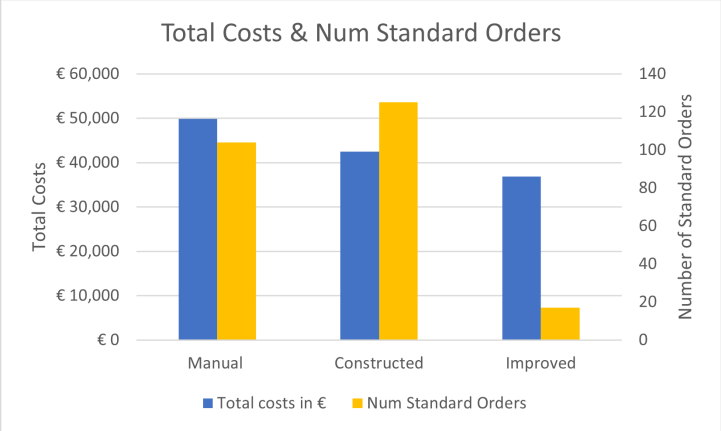


Figure 35: Comparison of schedules on the number of standard orders and total costs showing that the lowest total costs can be reached with only a fraction of the standard orders used.

5.9 Solution Robustness

From the results of all experiments performed, we found that the best performing setting of the grid search (Section 5.6) did not find a solution with lower costs than the experiment 5 of the success rate experiment (Section 5.5). This shows that luck with the randomness plays a role in finding the schedule with the lowest Total Costs. In this section, we will investigate this role to be able to give a more robust advice to Company X on the performance of the improvement algorithm. Section 5.9.1 introduces and explains an extension to the improvement algorithm to improve the exploration of the procedure. In Section 5.9.2, we experiment with randomness in the improvement algorithm and interpret their effects.

5.9.1 Random Repair

The ALNS presented in Section 4.4 uses four heuristics to create a new solution from the current solution. The two repair heuristics have a similar goal, to find the most cost-effective order to insert. However, it can be beneficial to use an explorative heuristic as a repair heuristic to escape local optima. This reduces the possibility that a procedure starts in a bad local optima and cannot escape from these solutions. A random repair heuristic is suitable for this use-case and is easy to understand and implement. The idea of the random repair is that it picks a random order to insert into a random vehicle. To do this for only feasible insertions, we first check for each vehicle whether an unplanned order is feasible within the vehicle. These feasible insertions are added to a list from which an insertion can be picked at random. After an insertion is chosen and implemented in the solution, the insertion list is updated to remove unfeasible insertions and change insertions into the vehicle which contains a new order. This process is repeated until there are no more feasible insertions in the list. This stopping criterion is the main difference with the existing repair heuristics, which stop if the insertions are not cost-effective, making the random repair more explorative than the greedy and regret repair heuristics. The technical description of the random repair can be found in Appendix G.

5.9.2 Solution Stability

With the random repair heuristic added to the ALNS, we will experiment with the robustness of the solutions. We repeat two experiments settings multiple times with different random seed values to compare the results. We will compare the best scoring experiment of this chapter with a Total Costs of €36.894 and the best performing setting of the grid search with a Total Costs of €37.434. The experiment

settings and result summary are shown in Table 19 and 20 respectively. The results per experiment are shown in Appendix H.

Table 19: Robustness Experiment Settings

Experiment Name	**Update Criterion**	**Degree of Destruction**	**Success Rate**	**Roulette Wheel Parameter**
Lowest Costs Setting	3	15% flat	[70,23,4]	0.1
Best Grid Search Setting	3	20%-10% decreasing	[88,29,5]	0.3

Table 20: Robustness Experiment Result Summary

Variable	**Low Cost setting**	**Grid Search setting**
Mean (μ)	€38.143	€38.382
St.dev (σ)	€628	€738
CI lower bound	€37.883	€38.078
CI upper bound	€38.402	€38.687
MaxCIHW	€120	€120
Expected n^*	55	75

The results show for both samples that the solution values are indeed dependent on the seed values. The confidence intervals ($\alpha = 5\%$) for the lowest cost and grid search experiment are [€37.883, €38.402] and [€38.078, €38.687] respectively. The calculations of this section can be found in Appendix H.1. Since the values of the confidence intervals overlap, we could not say statistically that there is a statistical difference between the groups, although we found a lower mean for the Low Cost setting. To create non-overlapping confidence intervals of mean, we would need to reduce the width of both intervals. To create sufficiently small intervals, we require approximately 130 runs in total. With the current sample sizes, we can still compare the results using a two sample t-test on the sample mean. Therefore, we have to assume normality, a reasonable assumption because of Central Limit Theorem (CLT) since the sample sizes are equal or larger than 25. The two tailed p-value from this test is 0.22 (Appendix H.2). With this p-value, we cannot reject the hypothesis of equal means (H_o) with $\alpha = 5\%$, which means that the data did not provide enough evidence that the sample means are different. Despite having no statistical significance, we recommend using the Lowest Cost setting above the Grid Search setting because of the found difference in mean of €240 and the lower standard deviation of the sample.

5.10 Conclusion

In this section, we investigated the settings for the ALNS. Therefore, we had to define the performance measures enabling comparison of the results. Total Costs is the most suitable performance measure to compare experiments and schedules in general because it balances the cost of not serving pallets and driving and working according to the assumptions of Section 4.5. If the costs of experiments are very similar, we can also use general ALNS measures, namely the acceptance ratio and the number of global best solutions found.

In the different experiments sets of Sections 5.3-5.6 we found sufficient differences in solution values to draw conclusions on the performance of different settings. For the degree of destruction input parameter, which determines how much of the current solution will be destroyed, we found the best solution with a decreasing interval between 10%-20% of customers. The success rates σ evaluate the heuristics on their success. The best performance is found with high values of σ and large differences between the different values $\sigma_1, \sigma_2, \sigma_3$. For the roulette wheel parameter ρ, a parameter for the adaptability of the algorithm, a high value yields the best performance. Nevertheless, the best solution found in all experiments executed does not have the largest σ and ρ tested. To make a more robust conclusion, we experimented with different seed values on the Low Cost setting (experiment 5 of Section 5.5) and best performing Grid

Search setting (Experiment 8 of Section 5.6). We concluded the lowest cost setting outperformed the best performing grid search experiment setting over multiple runs. The confidence interval of this setting is [€37.883, €38.402] ($\alpha = 5\%$). However, the confidence intervals of the means overlap and the p-value of the two-sample t-test is 0.22 and therefore is not significant with $\alpha = 5\%$. Nevertheless, we recommend the setting of the lowest cost experiment to Company X, because of the lower sample mean(€240) and the lower sample standard deviation. These settings are shown in Table 21

Table 21: Recommended Solution Input Parameters

Update Criterion	**Degree of Destruction**	**Success Rate Parameters**	**Roulette Wheel Parameter**
3	15% Flat	[70,23,4]	0.1

Within the ALNS we used 4 different heuristics to create new solutions (see Sections 4.4.2-4.4.5). From an analysis of the heuristics, we found that greedy removal is slightly more used than random destruction (55% and 45% respectively). Greedy repair is more used and has more success than regret repair, mainly because of the success of greedy repair in the first few iterations, it achieves a very high probability to be picked as repair heuristics. In combination with high values of sigma and good success in finding global and current best solutions, it retains a large pick probability for a significant time within the experiments.

The created schedules with the constructive and with the improvement algorithm both find schedules with lower total costs (€42.429 and €36.894 respectively) compared to the manual planning (€49.903). The relative difference is 15% and 26%. The robustness experiment showed an average solution value of €38.142 on this setting of the improvement algorithm, which would still result in a relative difference of 24%. The manual planning has the most unused capacity in the vehicles. It has 62 unused pallet places (7.5%) whereas the constructive and improvement algorithm have 17(2.1%) and 4(0.5%) respectively. The manual schedule also suffers from overplanning, scheduling more pallets to a vehicle than there is capacity, and therefore making on average 1 stop more than the constructive or improvement algorithm schedules. The reason for this may not only be because of suboptimal scheduling by the planning personnel. The difference in performance may also be caused by simplifications or assumptions of practical factors which are or cannot be modelled. The comparison of constructive and improvement algorithm show that the usage of most standard orders does not result in the solution with the lowest costs. Therefore, the usage of most of the standard orders is not cost-effective. The main reasons for this conclusion are rare updates of this set and a general reduction in size of orders of customers in this set (see Section 2.1.1). Overall, the constructive and improvement heuristic show that cost reductions can be achieved within the context and data provided by Company X.

6 Implementation

In this chapter, we present how the solution design of 4 can be used in practice for the scheduling department of Company X. Section 1.3 indicates that implementation are often not described in literature and therefore cause a gap between theory and practice. The implementation is twofold. Firstly, in Section 6.1, we describe the architecture of our solution and how this is can be accessed. Section 6.2 discusses how both of our solution designs could be used in practise by planners for finding a solution to their preference. We end the chapter by discussing the application possibilities in other sectors than the field in which Company X is operating (Section 6.3).

6.1 Architecture

In order to design a solution with a lot of complicating factors (e.g. trailers, cool units) a simple architecture to model the problem is not possible. Therefore, we designed an object based architecture in python 3.8.5. We explain the designed architecture in Section 6.1.1. Afterwards, in Section 6.1.2 we discuss how this architecture is usable in applications such as a decision support system.

6.1.1 Internal Architecture

The problem context of Company X contains multiple complex concepts, for example, cool units and trailers. A technical architecture helps to translate this concepts to a working solution solving environment in python (or any other programming environment). Additional benefits of a clear architecture are that it allows for easier maintainability as well as possibilities for others to change the methods in case of changes in the future. The architecture evolves around the Problem-Instance object (see Figure 36). This class contain multiple input attributes which can be other objects or general inputs, for example numbers or strings. The other objects are defined around the Problem-Instance object and the relation with the object is indicated with different entity relationship arrows. For example, the Fleet object requires one or many objects of the type Vehicle. The order object is one of the most complex objects which contains of one Location and one or many Load objects. Descriptions of the input attributes are found in Appendix J.1.

The solving process happens with the class functions of Problem-Instance object. In this process attributes are added to each of the object which store information about the current state of the solving process. For example, the OrderList of a vehicle stores which orders are driven by that vehicle in the current state. These attributes allow for the creation of the schedule and to translate the objects and attributes back to an interpretable schedule. The definitions of these attributes are found in Appendix J.2.

6.1.2 Architecture Access

In order to use the architecture in any system outside a python environment, a universal mapping of the architecture is required. Therefore, we created JSON mapping which can be used to translate & transfer (map) data from your data environment into the internal architecture previously explained. JSON is a text-based format for representing data structurally. We choose for a JSON because it is widely known and easy to use. Therefore, it allows to be used for other customers of the CAPE groep as well. This mapping is shown in Appendix I.1. It consists out of the input attributes of Figure 36. After solving the problem, the results have to be returned to your data environment. Therefore, we created an import mapping (see Appendix I.2). This mapping contains the same elements as the export mapping with the addition of generated attributes. These attributes contain all information to extract the schedule manually in an environment of choice. However, we also create the solution attribute presenting the schedule comprehensively. There is a possibility to export this comprehensive schedule to excel as well.

6.2 Solution Usage

In Section 5.8 we have seen that using our created solution can provide feasible solutions of better quality on various performance indicators compared to the manual schedule. However, this is given the boundaries in which the solving methods operates. As explained in section 4.5, various factors in practise are not implemented at all or simplified. Therefore, even the optimal schedule could not be used in practice. In this section, we will provide insights/instructions in how a planner could use both solving methods in

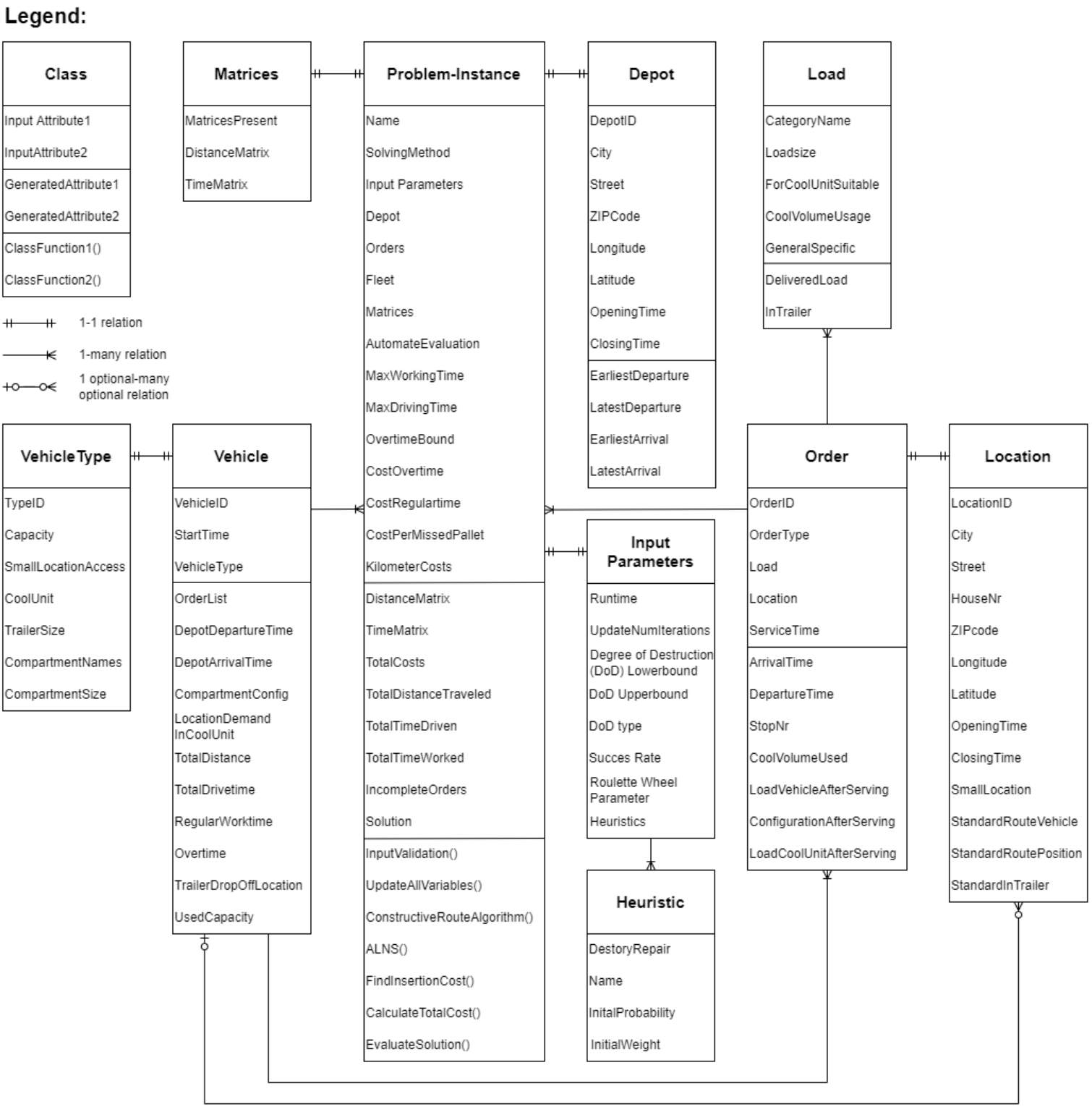


Figure 36: Internal Technical Architecture Diagram

order to create the best schedule given the variables in the model and reality. Testing in practice, whether these method result in improvements are not achievable in the limited time of this research. Nevertheless sharing these decision structures are valuable for future research. We describe two practical scenarios in which both algorithms are used to find a solution to two separate problems.

6.2.1 Schedule from scratch

Creating a schedule from scratch means that there is often more time to create a schedule than a few minutes. Therefore, it may seem beneficial to run the improvement algorithm for a longer time and only manually adjust some practical unpreferred stops. However, this has three disadvantages. First, it is impossible for a planner to know upfront the number of changes required for the best schedule. Consequently, the planner has to account for sufficient time after finishing, when determining the runtime of the improvement heuristic. Secondly, if the schedule is rejected completely, there may not be sufficient time to make a new schedule with different settings or inputs. Thirdly, inputs may unexpectedly change during the execution of the procedure and the planning has to start over from scratch, since the results became irrelevant. Therefore, we suggest the following iterative approach. This approach is visualized in the flow chart of Figure 37 and explained in the remainder of this section.

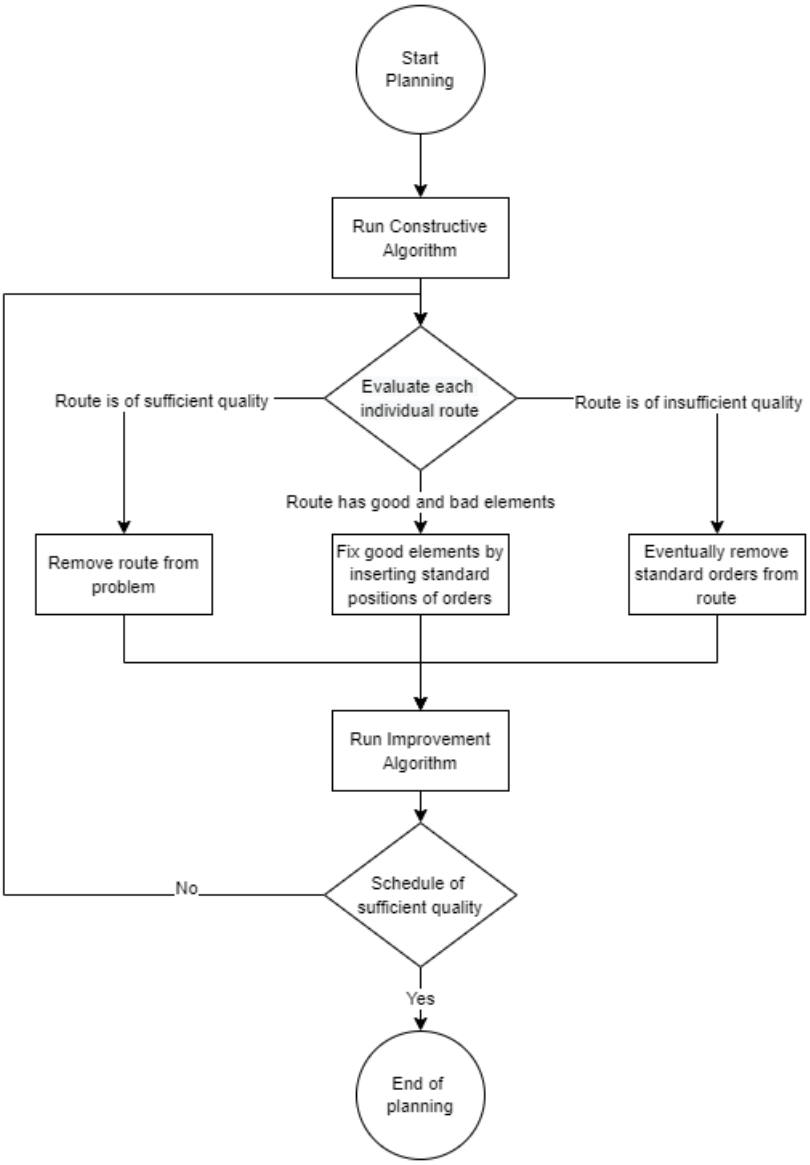


Figure 37: Planning approach to create a schedule from scratch

The essence of the decision structures is to create the schedule iteratively instead of running the improve-

ment algorithm for a longer period. To improve the outcomes of subsequent iteration the evaluation is necessary, otherwise the input is the same and the outcome will be the same as well. When evaluating a route manually, there are in general three possible evaluations. The route is of sufficient quality is the most simple and convenient outcome. Then, the route is definite and the orders and vehicle can be removed from the problem. The second option is a route is of insufficient quality. This outcome is more difficult to influence, the only possibility is to remove standard orders from the vehicle. The last possible evaluation outcome is that the route has good and bad elements. Then fixing these elements by applying standard vehicle and positions to this order is advisable. When a planner is evaluating a route and the route is for example sufficient if two orders positions are swapped. Then the planner should apply this changes manually and then evaluate the changed route as sufficient.

With a smaller problem and fewer clues on what is good or not in a schedule, it is more effective to run the improvement algorithm with a larger degree of destruction, since you may want more radical different solutions to the problem. However, this can only be proven beneficial in a practical environment. Another important input for the improvement algorithm is the runtime. If the evaluation phase takes longer then expected, the runtime has to change as well to be able to do enough iterations or to finish scheduling on time at all. Therefore, it is important to also think ahead about the time available and the desired number of iterations remaining in the process. The runtime can be reduced when the problem size is reduced to remain a similar number of iterations within the algorithm.

This approach allows for a more flexible way of creating a schedule from scratch compared to a single run of the improvement algorithm. A possible performance benefit may be achieved because of cooperative manual and automated actions. This can be easily tested in practice, by running the improvement algorithm parallel on a separate computer to check the difference between the two schedules. One of the conclusions could be that you considered some routes too early as perfect and that the improvement algorithm found a better configuration. This then can be used as standard vehicle and position input for a next schedule. With this decision structure, planners and the algorithm learn simultaneously from each other.

6.2.2 New schedule unavailable vehicle

Another plausible planning scenario is when a vehicle becomes unavailable and the schedule has to be changed. An approach of this scenario is visualized in Figure 38. A complete change of the planning is undesirable since most if not all vehicle are already loaded and changing all load is not possible. Therefore, select only a few vehicles and orders in the surrounding of the route of the unavailable vehicle. Because of the greatly reduced problem size, the improvement heuristic must be able to do multiple iterations and thus find a good solution given the standard input parameters, even with a smaller run time. When an instant solution is desired, possible to use the improvement heuristic as a more precise constructive heuristic on the principles of the repair method. To do this, set the degree of destruction to the problem size, so that it will repair and empty solution and only consider your preferred repair heuristic. An experiment must determine which repair heuristic is most suitable for this scenario. The runtime must be set to 0 to perform only 1 iteration. With one run, a sufficient schedule should be produced to deal as good as possible with the loss of a vehicle. The actual runtime will not 0 logically, but will be marginalized with a small problem subset.

A practice test has not been performed because of time limitations. We hypothesize that this approach mostly saves time and effort rather than costs. The reason for this is that a planner can create good clusters of orders out of the unavailable vehicle, and its close neighbours. With these clusters, it is possible to create cost-efficient routes. However, finding the right order within clusters is most time-consuming, whereas the suggest approach can do this instantly. With a longer runtime, costs savings are achievable because of evaluation of more neighbourhoods and deeper search within those neighbourhoods.

6.3 Broad Application

Vehicle scheduling problems have a much broader application than the specific context of Company X. For example, a similar problem context is the transport in the agricultural sector. In this field, trucks with cooled compartments for fresh food are not uncommon. Even frozen transport compartments are possible within the technical architecture. For CAPE groep it may be valuable to known whether the architecture can be applied by different customers even in other sectors. Every sector has different quirks, but the

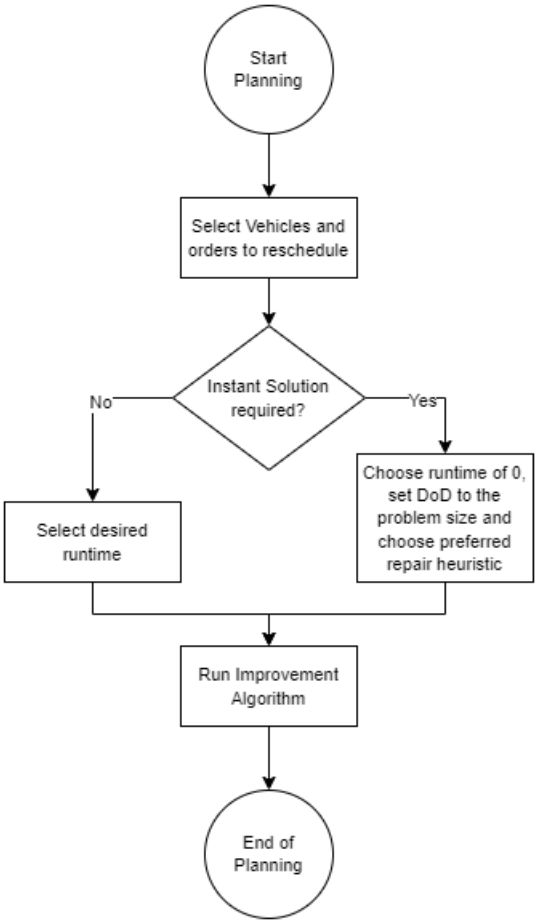


Figure 38: Planning approach to optimize the planning when a vehicle becomes unavailable

general architecture presented can apply in a lot of more complex routing problems. This is possible because some variables used in the architecture of Section 6.1 do not have a fixed unit. In the remainder of the section, we discuss other application areas in which our solution could be used and how this can be applied.

Liquid Transport
The presented architecture is capable of handling liquid transport as well, especially if those transport cannot be mixed. For example, a distributor of petrol and diesel. Some of their vehicles may have two liquid tanks, but some of them may have only one larger tank. This can be created by changing the compartment size attribute value. For example, a vehicle with two tanks of 1000L can be modelled with a vehicle with capacity: 2000; Compartment names ["Diesel","Petrol"] and a compartment size of 1000. This vehicle can be filled with 2000L diesel or petrol, or 1000L of both types. It is even possible to add more liquids to the compartment names. The algorithm will decide which liquid type is assigned to a tank and thus which liquids will be loaded in the vehicle. In a multiple liquid type case, it is advisable to split loads of customers into different orders since the architecture does not allow for split delivery of orders, which is more logical in this field than in the field of Company X. The same changes can be made for another possible transportation sector, namely the transportation of raw material such as grain/corn and coal/lignite.

Person Transport
The transportation of persons is another category of transport in which the architecture can be applied. The solving algorithm is capable of handling direct orders. Orders in which the pickup and delivery are both not the starting depot. In literature, this is referred to as the Dial A Ride Problem (DARP). The most common application is the door-to-door transport of elderly or disabled people (Cordeau and Laporte, 2007). In this case the capacity and Loadsize should be both set to one and most of the features as CoolUnit and TrailerSize can be disabled.

A sub-category of DARP is the School Bus Routing Problem (SBRP). In this problem, a number of students have to be picked up and delivered to school. Applying the architecture to the SBRP is done similar as the DARP except that the capacity of the vehicle is much larger. The closing of time of the school location can be set to the latest arrival at school to ensure all students are on time at the school. With this method, the start time of the driver can be reasonable accurately determined.

6.4 Conclusion

This section explained how an implementation of our solution explained in Chapter 4 can be used in a practical environment. For modelling complicating concepts (e.g. trailers & cool units) an object based architecture is created enabling the algorithm to solve different problem instances. This type of architecture has increased maintainability and adaptability because of the defined relations and dependencies between objects. Obtaining a solution with the architecture is done by mapping data into the desired format with JSON, a text based data structure format. When one of the algorithms produced a schedule, it can be retrieved by an import mapping. To support this process and the understandability of all attributes, explanation of all attributes are provided in the appendices.

A schedule created by our solution designs may be not usable into practice because of the assumptions, limitations and simplifications. Therefore, we described two ways how planners can use our solution designs to their advantages in different situations. The first situation describes a scheduling from scratch procedure in which the constructive or improvement algorithm are combined with manual evaluation of routes. The planner judges whether routes are of sufficient quality, satisfactory routes are fixed. With the remaining routes, repeating iterations are performed until all routes are judged of sufficient quality by the planner. The second situation describes a quick repair of a schedule in which a vehicle cannot be used. For this situation, the improvement algorithm is used on a problem subset with a reduced runtime. We expect this procedure creates solutions of at least similar quality compared with planners in a fraction of time.

In the last part of this chapter, we discussed the possibilities of using our solution design in a broader context than in the field of medical supply transport. In the field of liquid transport or person transport, the provided architecture of Section 6.1 could be applied as well. This is possible because some variables

in the model do not have a fixed unit. Therefore, the designed solution allows for a broader usage outside the problem context of Company X.

7 Conclusion, Discussion and Recommendations

This final chapter of our research shows the conclusions in Section 7.1. Section 7.2 discusses the contribution and limitations of the study. Lastly, Section 7.3 gives recommendations to the company as well as suggestion for further research.

7.1 Conclusion

In Chapter 1, we formulated the following main research question: "How can a decision support system for transport route scheduling help Company X's planning personnel to create schedules faster and more efficient with lower routing costs?" We analysed Company X's multistep planning process to schedule their fleet of ± 45 trucks and ± 5.000 orders a day and concluded that the planning step which assigns distribution orders to vehicles is the largest contributor to the problem. This is caused by the experience dependency of the step, the labour intensity and the fact that the recovery planning is executed by the same personnel after doing the distribution planning. Therefore, to solve the main research question, the solution should be able to create a distribution planning and help with making changes caused by unforeseen events (recovery planning). On top of that, the scheduling solution should be able to handle different order types and multiple complex constraints, as well as provide the planner with relevant performance measures.

In literature, all obligatory and desired constraints of the problem context of Company X were used studies. However, the full problem context is not applied in literature. Therefore, we defined a new routing problem, the Multi-Trip Multi-Compartment Pickup and Delivery Problem Time Windows (MTM-CPDPTW). The most appropriate solution methods for this problem are heuristics and meta-heuristics. For a fast running solution, a savings algorithm suits the problem context best because of its ability to prioritize with limited capacity and its handling of infeasibility. For a more elaborate search for the best solution, the ALNS is preferred given the problem context because of adaptive approach of neighbourhoods and its methods to create new solutions.

With the order data provided by Company X, we tested the possibility to solve their problem with exact method. The dataset contains ± 3.000 orders, which represents an average day at Company X. From this test, we concluded that solving with an exact method is not computationally feasible in reasonable time. Therefore, we designed two solutions for fast and elaborate solving of the problem. The availability of a fast running (constructive algorithm) and elaborate solving method (improvement algorithm) allows for the personnel to balance faster and more efficiently scheduling with finding the lowest route costs. The constructive algorithm is a five steps procedure using predetermined orders fixed to vehicles to create a schedule fast. Firstly, all fixed positioned predetermined orders are inserted into their corresponding vehicle. Secondly, the non fixed orders are inserted. The third step is creating the set of orders with limited accessible locations to serve. Fourthly, these "small" orders are inserted into their most profitable vehicle. Lastly, as many as possible remaining orders are inserted with an adaptation of a greedy saving algorithm. The improvement algorithm is an ALNS tailored to the problem context of Company X. It uses random and greedy destroy heuristics together with a greedy and repair heuristics to generate neighbouring solutions.

To compare the schedules with each other, we use the total cost as the performance measure. Total costs represent most accurately complete scheduling costs, because it balances the cost of not serving pallets and driving and working. From experimenting with different input settings, we found that randomness plays a role in the performance of certain settings. Therefore, we performed an additional experiment on the robustness of the solution, by testing different random seeds. We found that a setting in an earlier experiment with low costs (Low Cost setting) outperformed the best setting of the grid search (Grid Search setting). However, the 95% confidence intervals of the samples overlap [€37.883, €38.402] and [€38.078, €38.687] for the Low Cost setting and Grid Search setting, respectively. A two-sample t-test, with 25 samples for both settings, showed a p-value of 0.22. The confidence interval comparison and the t-test do not provide statistical evidence that the sample means are significantly different. Nevertheless, the experiment showed a lower mean and lower standard deviation for the Low Cost setting, and therefore recommend this setting. The parameter values of this setting are: *Update Criterion = 3; Degree of Destruction = 15% flat; Success Rates = [70,23,4] and a Roulette Wheel Parameter ρ = 0.1.* The designed solution produce schedules with lower total costs. The constructive algorithm (€42.429) saves 15% compared to the manual schedule (€49.903). Whereas, the savings of the improvement algorithm

(€36.894) are up to 26% compared to the manual schedule. On average, this saving is 24%. From these differences, we conclude that significant costs reductions are achieved by using the designed solutions. Another finding is that the usage of most standard orders is not cost-efficient for the schedule.

Implementation of the designed solutions is possible with the designed object-based architecture. The architecture can be accessed with a mapping of problem data in a data environment into the architecture. This makes the solution implementable in a decision support system. Practical usage can be limited by the modelling assumptions and simplifications. Therefore, we designed two decision structures which combine manual and automated actions. One decision structure is for building schedules from scratch, the other structure is for repairing a solution with becomes infeasible because of capacity reduction. The way our solution is structured allows for a broader application than the problem context of Company X. Other transport fields of routing problems which can apply our solution are the liquid transport and the person transport sector.

7.2 Contribution & Discussion

The contribution and discussion of this research is twofold, practical and academic. For both categories, we discuss how this research contributed practically or academically and discuss limitations in our study. Starting with the practical contribution & discussion in Section 7.2.1 and ending with the academic contribution & discussion in Section 7.2.2.

7.2.1 Practical

The main practical contribution of this research is the development of two design solution which allow Company X to save up to 26% on their routing costs, with an average of 24%. The design of these solutions is supported with a data architecture and JSON mappings allowing implementation within a system, including the OOMHPA currently developed for Company X. This is further supported with decision structures for the planners to use the solution design in a practical environment. The second contribution for Company X is an analysis on the usefulness of standard orders. A standard order is an order which is assigned to a specific vehicle by default (see Section 2.1.1). This can help Company X by evaluating whether standard orders derived from the base planning are still a useful part in their planning process. The last practical contribution is the generalisability of the solution designs in combination with the data architecture description and the descriptions of the constructive algorithm and improvement algorithm with all used heuristics. This allows researchers to develop their own version of the solution design for routing problems. These routing problems do not have to be similar to the problem of Company X as shown in Section 6.3.

The most important discussion point of the practical contributions is the limited time to test the solutions designed in practice with the planning personnel of Company X. This makes it difficult to validate whether time and effort savings can be achieved by using the algorithms in the described method of Section 6.2 or in another way. Our hypothesis is that our solution provides time and effort savings, because developing routes from scratch is more difficult than evaluating a methodical constructed routes. Even if these constructed routes require some changes for practical feasibility. Nevertheless, evaluating a schedule and individual routes (for example, on work time, travel distance and total costs) becomes easier with our implementation because of the provide evaluation attributes.

7.2.2 Science

This research has multiple scientific contributions to the literature. Section 3.1.13 provides to the best of our knowledge the first definition of the Multi-Trip Multi-Compartment Pickup and Delivery Problem Time Windows (MTMCPDPTW). The theoretical formulation in Section 4.2 provides a starting point for other researchers in this complex, unsearched field of routing problems. Secondly, we have shown that the described 3-index formulation is not solvable exact in reasonable time for problems with more than six customers. The last contribution is that the solving the MTMCPDPTW with ALNS in the context of Company X requires significant computational power. When compared to ALNS literature solving PDPs, the number of iterations achieved is $\approx$100 times lower with 1 hour runtime. Further research is suggested to further investigate and try to reduce this difference.

There are two discussion points related to the academic contribution of our research. Firstly, we were

not able to perform the experiments on multiple datasets of Company X. This limits the validity and the robustness of the provided settings and its value to other researchers if they develop similar designs for their routing problems. Secondly, the number of iterations of the ALNS were ≈100 times less than the ALNS algorithms found in literature. The low number of iterations reduces the effectiveness of the adaptiveness of the procedure and increase the effect of randomness. Therefore, we defined multiple further research directions to increase the number of iterations.

7.3 Recommendations & Further Research

Based on our conclusions, we formulate several recommendations for Company X. We also defined directions for further research on the topic. Starting with two practical context focused recommendations, followed by four further research suggestions.

The first recommendation for Company X is to further investigate the effectiveness of standard orders in their scheduling process. This research has shown that better schedules can be made without the use of 86% of the standard orders. This means that a large proportion of the standard orders is suboptimal. The two main identified reasons for this are too few updates on the base planning and a reduction in volume of standard orders. Therefore, we recommend investigating which standard orders are still viable and afterwards to regularly evaluate their effectiveness and make changes accordingly.

The second recommendation is to perform tests in a practical controlled environment. The step from the current planning process to an automated planning solution is difficult to execute. We recommend a test in practice using the designed decision structures. This can increase the support and engagement for a change of the planning process by all stakeholders. An additional benefit is that the optimal usage of both solution designs can be found and the decision structures of Section 6.2 can be evaluated.

The first suggestion for further research is the exploration of the MTMCPDPTW. This research is the first to define this routing problem and describing a problem formulation. However, the 3-index formulation suffers from symmetry and a weak linear relaxation (Section 4.2.2). One of the option to mitigate these effects is the usage of column generation techniques. Applying column generation to this problem context may result in the possibility to solve this problem exact in reasonable time.

The second suggestion is closely related to the practical tests recommended before, is to test the effectiveness of the recommended settings on multiple datasets. This research only tested the performance on one dataset, however the results may vary on other datasets. Therefore, we suggest researching the performance of the recommended settings of the improvement algorithm on other datasets.

Thirdly, we propose further research on the heuristics used in the ALNS which are more specifically tailored to the PDP problems or even the specific problem context of Company X. Due to the limited time available within this study, we were unable to put additional effort into developing more heuristics other than the two destroy and repair heuristics developed and tested.

Fourthly, we suggest researching and developing ways to reduce the time per iteration for the improvement algorithm. The current version of the algorithm spends multiple seconds to run one iteration, severely limiting the number of iterations per run. There are two options to decrease the number of second spend per iteration, which do not change the algorithms itself. The first option is building the architecture and algorithms in a different programming environment. Python is a programming language that makes developing applications easier and more pleasant compared to other environments. However, this comes with the downside of longer runtimes for these applications. Although there are code optimization possible in python, they do not always comply with an object based architecture. The second option is to use a faster, up-to-date system to run the algorithms. The system used for testing does not have the largest processing capacity, and therefore a recent high-end system can improve performance significantly. These two methods can only partly solve the computational complexity of the algorithms. Therefore, we have three research suggestions to improve the runtime with different difficulties and impacts. Those are listed below.

- Investigate options to run an ALNS for the MTMCPDPTW in parallel.
  - Parallel solving methods use multiple processing cores (or computers) to solve a problem,

and therefore can solve more complex problems in less time. However, it is more difficult to program and has more costs in developing and using the solving environment.

- Reduce checked orders for insertion in the repair heuristics
  - Within the repair heuristics, all unplanned orders are checked for insertion in vehicles. This set could be reduced randomly, or by logic, for example only allowing insertion of geographically close-by orders in vehicles. Both result in a reduction of a neighbourhood size per iteration, and thus the computation time per iteration.
- Estimate insertion costs by repairs
  - Currently, all possible insertions (in vehicles with sufficient free capacity) of orders are calculated to determine the insertion costs. Instead of exactly calculating the insertion costs, these could also be estimated by indicators of the costs, such as delivered load and the additional required drive/work time and additional distance.

References

L. Abbatecola, M. P. Fanti, A. M. Mangini, and W. Ukovich. A decision support approach for postal delivery and waste collection services. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, 13(4):1458–1470, 2016. doi: 10.1109/TASE.2016.2570121.

T. J. Ai and V. Kachitvichyanukul. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5):1693–1702, 2009. doi: https://doi.org/10.1016/j.cor.2008.04.003.

R. Baldacci, N. Christofides, and A. Mingozzi. On an integer program for a delivery problem. *Math*, 12 (2):300–304, 2008. doi: https://www.jstor.org/stable/167930.

M. L. Balinski and R. E. Quandt. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Math. Program.*, 115:351–385, 2008. doi: https://doi.org/10.1007/s10107-007-0178-5.

E. Berhan, B. Beshah, D. Kitaw, and A. Abraham. Stochastic vehicle routing problem: A literature survey. *Journal of Information & Knowledge Management*, 13(3):1450022, 2014. doi: https://doi.org/10.1142/S0219649214500221.

K. Braekers, K. Ramaekers, and I. V. Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016. doi: https://doi.org/10.1016/j.cie.2015.12.007.

A. G. Canen and L. G. Scott. Bridging theory and practice in vrp. *The Journal of the Operational Research Society*, 46(1):1–8, 1995. doi: https://www.jstor.org/stable/2583830.

T. Caric and J. Fosin. Using congestion zones for solving the time dependent vehicle routing problem. *Promet-Traffic & Transportation*, 32(1):25–38, 2020. doi: http://dx.doi.org/10.7307/ptt.v32i1.3296.

D. Cattaruzza1, N. Absi, and D. Feillet. Vehicle routing problems with multiple trips. *4OR-Q J Oper Res*, 14:223–259, 2016. doi: https://doi.org/10.1007/s10288-016-0306-2.

E. D. Chajakis and M. Guignard. Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization*, 26:43–78, 2003. doi: https://doi.org/10.1023/A:1023067016014.

W. Chowmalia and S. Sukto. A hybrid fja-alns algorithm for solving the multi-compartment vehicle routing problem with a heterogeneous fleet of vehicles for the fuel delivery problem. *Decision Science Letters*, 10:497–510, 2021. doi: doi:10.5267/j.dsl.2021.6.001.

C.-W. Chu. A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, 165(3):657–667, 2005. doi: https://doi.org/10.1016/j.ejor.2003.08.067.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964. doi: https://doi.org/10.1287/opre.12.4.568.

J. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Ann Oper Res*, 153:29–46, 2007. doi: https://doi.org/10.1007/s10479-007-0170-8.

G. B. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959. doi: https://doi.org/10.1287/mnsc.6.1.80.

U. Emeç, B. Çatay, and B. Bozkaya. An adaptive large neighborhood search for an e-grocery delivery routing problem. *Computers & Operations Research*, 69:109–125, 2016. doi: https://doi.org/10.1016/j.cor.2015.11.008.

R. Eshtehadi, E. Demir, and Y. Huang. Solving the vehicle routing problem with multi-compartment vehicles for city logistics. *Computers & Operations Research*, 115:104859, 2020. doi: https://doi.org/10.1016/j.cor.2019.104859.

S. Few. Dashboard confusion. *Intelligent Enterprise*, 2004.

S. Few. *Information Dashboard Design*. O'Reilly, 2006.

B. Fleischmann. The vehicle routing problem with multiple use of vehicles. Working Paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Germany, 1990.

M. G. S. Furtado, P. Munari, and R. Morabito. Pickup and delivery problem with time windows: A new compact two-index formulation. *Operations Research Letters*, 45(4):334–341, 2017. doi: https://doi.org/10.1016/j.orl.2017.04.013.

M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996. doi: https://doi.org/10.1016/0377-2217(95)00050-X.

M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174, 2006. doi: https://doi.org/10.1016/j.trc.2006.03.002.

M. Gendreau, J. Y. Potvin, O. Bräumlaysy, G. Hasle, and A. Løkketangen. *Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography*, pages 143–169. Springer, Boston, 2008.

B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974. doi: https://doi.org/10.1287/opre.22.2.340.

M. Goetschalckx and C. Jacobs-Blecha. The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42:39–51, 1989. doi: https://doi.org/10.1016/0377-2217(89)90057-X.

A. Gutierrez, L. Dieulle, N. Labadie, and N. Velasco. A multi-population algorithm to solve the vrp with stochastic service and travel times. *Computers & Industrial Engineering*, 125:144–156, 2018. doi: https://doi.org/10.1016/j.cie.2018.07.042.

A. Gutiérrez-Sanchez and L. B. Rocha-Medina. Vrp variants applicable to collecting donations and similar problems: A taxonomic review. *Computers & Industiral Engineering*, 164:107877, 2022. doi: https://doi.org/10.1016/j.cie.2021.107887.

H. Heerkens and A. van Winden. *Solving Managerial Problems Systematically.* Noordhof Uitgevers, 2017.

K. Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000. doi: https://doi.org/10.1016/S0377-2217(99)00284-2.

H. H. Hoos and T. Stützle. 9 - scheduling problems. In H. H. Hoos and T. Stützle, editors, *Stochastic Local Search*, The Morgan Kaufmann Series in Artificial Intelligence, pages 417–465. Morgan Kaufmann, San Francisco, 2005. ISBN 978-1-55860-872-6. doi: https://doi.org/10.1016/B978-155860872-6/50026-3.

A. Hübner and M. Ostermeier. A multi-compartment vehicle routing problem with loading and unloading costs. *Transportation Science*, 53(1):282–300, 2018. doi: https://doi.org/10.1287/trsc.2017.0775.

S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003. doi: https://doi.org/10.1016/S0377-2217(02)00147-9.

M. S. Innocente. Population-based methods: Particle swarm optimization – development of a general-purpose optimizer and applications –. Master's thesis, Universitat Politècnica de Catalunya, Spain, 2006.

M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989. doi: https://doi.org/10.2307/2289924.

Y.-H. Jia, Y. Mei, and M. Zhang. A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem. *IEEE Transactions on Cybernetics*, 28(10):1–14, 2021. doi: 10.1109/TCYB.2021.3069942.

H. Jula, M. Dessouky, and P. A. Ioannou. Real-time estimation of travel times along the arcs and arrival times at the nodes of dynamic stochastic networks. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):97–110, 2008. doi: https://doi.org/10.1109/TITS.2007.908571.

P. Karnick, D. Cline, S. Jeschke, A. Razdan, and P. Wonka. Route visualization using detail lenses. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):235–247, 2010. doi: 10.1109/TVCG.2009.65.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995. doi: 10.1109/ICNN.1995.488968.

J. Kennedy and R. Eberhart. The particle swarm: social adaptation of knowledge. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, pages 303–308, 1997. doi: 10.1109/ICEC.1997.592326.

P. N. Ky Phuc and N. L. Phuong Thao. Ant colony optimization for multiple pickup and multiple delivery vehicle routing problem with time window and heterogeneous fleets. *Logistics*, 5(2):28, 2021. doi: 10.3390/logistics5020028.

P. Lacomme, G. Rault, and M. Sevaux. Integrated decision support system for rich vehicle routing problems. *Expert Systems with Applications*, 178:114998, 2021. doi: https://doi.org/10.1016/j.eswa.2021.114998.

J. Lenstra and A. R. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. doi: https://doi.org/10.1002/net.3230110211.

H. Li, H. Wang, J. Chen, and M. Bai. Two-echelon vehicle routing problem with time windows and mobile satellites. *Transportation Research Part B: Methodological*, 138:179–201, 2020. doi: https://doi.org/10.1016/j.trb.2020.05.010.

J. Li, Y. Ma, Z. C. Ruize Gao, A. Lim, W. Song, and J. Zhang. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. *IEEE Transactions on Cybernetics*, pages 1–14, 2021. doi: 10.1109/TCYB.2021.3111082.

Y. Li, H. Soleimani, and M. Zohal. An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives. *Journal of Cleaner Production*, 227:1161–1172, 2019. doi: https://doi.org/10.1016/j.jclepro.2019.03.185.

C. Lin, K. Choy, G. Ho, S. Chung, and H. Lam. Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, 41:1118–1138, 2014. doi: http://dx.doi.org/10.1016/j.eswa.2013.07.107.

R. Liu, Y. Tao, and X. Xie. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research*, 101:250–262, 2019. doi: https://doi.org/10.1016/j.cor.2018.08.002.

S. Majidi, S.-M. Hosseini-Motlagh, and J. Ignatius. Adaptive large neighborhood search heuristic for pollution-routing problem with simultaneous pickup and delivery. *Soft Comput*, 22:2851–2865, 2018. doi: https://doi.org/10.1007/s00500-017-2535-5.

S. Majidi, S.-M. Hosseini-Motlagh, and J. Ignatius. Adaptive large neighborhood search for the commodity constrained split delivery vrp. *Computers & Operations Research*, 112:104761, 2019. doi: https://doi.org/10.1016/j.cor.2019.07.019.

C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992. doi: https://doi.org/10.1287/trsc.26.3.185.

J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129, 2015. doi: http://dx.doi.org/10.1016/j.cie.2014.10.029.

I. Moon, J.-H. Lee, and J. Seong. Vehicle routing problem with time windows considering overtime and outsourcing vehicles. *Expert Systems with Applications*, 39:13202–13213, 2012. doi: https://doi.org/10.1016/j.eswa.2012.05.081.

L. Muyldermans and G. Pang. On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Juournal of Operational Research*, 206(1):93–103, 2010. doi: https://doi.org/10.1016/j.ejor.2010.02.020.

J. Oyola, H. Arntzen, and D. L. Woodruff. The stochastic vehicle routing problem, a literature review, part i: models. *EURO Journal on Transportation and Logistics*, 7(3):193–221, 2018. doi: https://doi.org/10.1007/s13676-016-0100-5.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58:81–117, 2008a. doi: https://doi.org/10.1007/s11301-008-0036-4.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58:21–51, 2008b. doi: https://doi.org/10.1007/s11301-008-0033-7.

D. Pisinger and S. Ropke. *Large Neighborhood Search*, pages 399–420. Springer, 2010. ISBN 987-1-4419-1663-1.

C. Prins. Efficient heuristics for the heterogeneous fleet multitrip vrp with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms*, 1(2):135–150, 2002. doi: https://doi.org/10.1023/A:1016516326823.

H. N. Psaraftis. *Dynamic Vehicle Routing Problems*, pages 223–248. North-Holland, 1988.

H. N. Psaraftis, M. Wen, and C. A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2015. doi: https://doi.org/10.1002/net.21628.

H. Pullen and M. Webb. A computer application to a transport scheduling problem. *Computer Journal*, 10(1):10–13, 1967. doi: https://doi.org/10.1093/comjnl/10.1.10.

S. Ropke and D. Pisinger. Adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006. doi: https://doi.org/10.1287/trsc.1050.0135.

D. Sacramento, D. Pisinger, and S. Ropke. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, 102:289–315, 2019. doi: https://doi.org/10.1016/j.trc.2019.02.018.

A. Santini, S. Ropke, and L. M. Hvattum. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *J Heuristics*, 24:783–815, 2018. doi: https://doi.org/10.1007/s10732-018-9377-x.

L. Santos, J. Coutinho-Rodrigues, and C. H. Antunes. A web spatial decision support system for vehicle routing using google maps. *Decision Support Systems*, 51:1–9, 2011. doi: http://dx.doi.org/10.1016/j.dss.2010.11.008.

A. Sarikaya, M. Correll, L. Bartram, M. Tory, and D. Fisher. What do we talk about when we talk about dashboards? *IEEE Transactions on Visualization and Computer Graphics*, 25(1):682–692, 2019. doi: 10.1109/TVCG.2018.2864903.

I. Sbai, S. Krichen, and O. Limam. Two meta-heuristics for solving the capacitated vehicle routing problem: the case of the tunisian post office. *Oper Res Int J*, 22:507–549, 2022. doi: https://doi.org/10.1007/s12351-019-00543-8.

K. Sidorov and A. Morozov. A review of approaches to modeling applied vehicle routing problems, 2021.

M. Sigurd, D. Pisinger, and M. Sig. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38(2):197–209, 2004. doi: https://doi.org/10.1287/trsc.1030.0053.

S.-Y. Tan and W.-C. Yeh. The vehicle routing problem: State-of-the-art classification and review. *applied sciences*, 11(21):10295, 2021. doi: https://doi.org/10.3390/app112110295.

C. Tarantilis and C. Kiranoudis. Using a spatial decision support system for solving the vehicle routing problem. *Information & Management*, 39:359–375, 2002. doi: http://dx.doi.org/10.1016/S0378-7206(01)00103-3.

F. Theurich, A. Fischer, and G. Scheithauer. A branch-and-bound approach for a vehicle routing problem with customer costs. *EURO Journal on Computational Optimization*, 9:100003, 2021. doi: https://doi.org/10.1016/j.ejco.2020.100003.

S. Voß, S. Martello, I. Osman, and C. Roucairol. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Springer US, 2012.

V. D. Wester. A genetic algorithm for the vehicle routing problem. Master's thesis, University of Tennessee, Knoxville, US, 1993.

T. Woensel, L.Kerbache, H.Peremans, and N.Vandaeled. Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research*, 186(3):990–1007, 2008. doi: https://doi.org/10.1016/j.ejor.2007.03.012.

Z. Yang, J.-P. van Osta, B. van Been, R. van Krevelen, R. van Klaveren, A. Stam, J. Kok, T. Bäck, and M. Emmerich. Dynamic vehicle routing with time windows in theory and practice. *Natural Computing: An International Journal*, 16(1):119–134, 2017. doi: https://doi.org/10.1007/s11047-016-9550-9.

I. Yusuf, M. S. Baba, and N. Iksan. Applied genetic algorithm for solving rich vrp. *Applied Artificial Intelligence*, 28(10):957–991, 2014. doi: https://doi.org/10.1080/08839514.2014.927680.

Y. Zhang, S. Wang, and G. Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015:38, 2015. doi: https://doi.org/10.1155/2015/931256.

Çagrı Koç and G. Laporte. Vehicle routing with backhauls: Review and research perspectives. *Computers and Operations Research*, 91:79–91, 2018. doi: https://doi.org/10.1016/j.cor.2017.11.003.

Appendices

This section contains the different appendices which are referenced throughout the document.

A Model Formulation

This Appendix will be structured in the following way: Section A.1 discusses the sets within the problem; Section A.2 discusses input parameters for the model; the variables which are decided by the model are discussed in Section A.3; Section A.4 discusses the objective of the model; Fifthly, in section A.5 constraints are discussed in related groups.

A.1 Model Sets

Sets describe the indices ranges for the parameters, decision variables and constraints. Larger sets will increase the problem size notably. Table 22 states all the sets with explanation and their range. A difference from the problem context within the formulation is the exclusion of a frozen temperature range, and item type. This is because in practice this item is seldom transported and therefore not worth modelling.

Table 22: Model Sets

Set name	**Explanation**	**Range**
Set of Vehicles K	Set to provide each vehicle with a index	$k \in \{0, 1, 2, ..., 35\}$
Set of Pickup Locations P	Set of locations which have goods for pickup (for distribution orders this is equal to the depot)	$i \in \{1, 2, 3, ..., 266\}$
Set of Delivery Locations D	Set of locations which require goods to be delivered (for collection orders this is equal to the depot)	$i \in \{227, 228, 229, ..., 532\}$
Set of Locations L	Union of the sets P & D, to describe all locations which have to be visited by vehicles	$i \in \{1, 2, 3, ..., 532\}$
Set of All Vertices A	A combination of the set of Location combined with 2 entries for the depot {0,453}	$i \in \{0, 1, 2, ..., 533\}$
Set of Compartment types C	Set which describes the compartment types within a vehicle {2-8°C, 15-25°C}	$c \in \{0, 1\}$
Set of Item Types G	Set which describes the different temperature ranges customer can order {2-8°C, 15-25°C,unspecified}	$g \in \{0, 1, 2\}$
Set of Configuration Sizes S	Describes the possible sizes of compartment in the main body of the vehicle	$s \in \{0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33\}$

* *In practice, the frozen temperature range does not occur often and therefore is not incorporated in the model*

A.2 Model Parameters

The model parameters are input for the model and their values cannot be changed when solving the model. Each parameter has a different size dependent on the indices of the parameter. For example, c^k is a list of parameters $\{c^0, c^1, ..., c^{35}\}$. The following list lists all parameters with an explanation. The indices are either sub-scripted if they relate to locations or items, or super-scripted if they relate to vehicles, compartments or configurations.

$q_{i,g}$ = Demand of customer i of item type g (pickup quantity is positive, delivery quantity negative)

c^k = Total capacity of vehicle k

$d_{i,j}$ = Distance from vertex i to vertex j

$t_{i,j}$ = Travel time from vertex i to vertex j

e_i = Earliest arrival time at vertex i

l_i = Latest departure time at vertex i

s_i = Service time at vertex i

mdt = Max driving time of a driver

ot = threshold to account drive time as overtime

lc_i^k = 1 if location i is accessible for vehicle k, 0 otherwise

w^s = Compartment size of configuration s

co = Costs in € per hour of work in overtime

cr = Costs in € per hour of work in regular time

cp = Costs in € per unfulfilled unit of demand

cq_i = 1 if cooled 2-8°C demand of location i is suitable for cool unit, 0 otherwise

cv_i = volume of cool unit used if demand of location i is placed in the cool unit

ca^k = 1 if cool unit is present in vehicle k, 0 otherwise

ts^k = Size of the trailer of vehicle k, 0 if no trailer is present

tl_i = 1 if trailer is suitable for location i, 0 otherwise

M = A large number for modelling Big M constraints

A.3 Model Decision Variables

The decision variables are chosen by the model to minimize or maximize the objective value of the model. In total their are 14 different categorized decision variables:

$X_{i,j}^{k}$ = 1, if vehicle k travels from vertex i to vertex j, 0 otherwise

$Q_{i,g}^{k}$ = Load of item g in vehicle k when arriving at vertex i

$L_{i}^{k,c}$ = The load of unspecified cargo in compartment c in vehicle k at location i

B_{i}^{k} = Beginning of service of vehicle k at vertex i

O^{k} = Number of hours driven in overtime of vehicle k

R^{k} = Number of hours driven in regular time of vehicle k

$C^{k,c}$ = Capacity allocated to compartment c in vehicle k

$Z^{s,c,k}$ = 1, if configuration s is chosen for compartment c in vehicle k, 0 otherwise

$Y_{i,g}$ = Number of demand of type g delivered at vertex i by the main vehicle body (pickup is positive, delivery negative)

U_{i}^{k} = 1, if cooled demand for vertex i is in cool unit of vehicle k, 0 otherwise

$D_{i,g}^{k}$ = Number of demand of type g delivered at vertex i by trailer of vehicle k (pickup is positive, delivery negative)

W_{i}^{k} = 1, if demand of vertex i is served form trailer of vehicle k

$T^{k,c}$ = 1, if trailer of vehicle k is of compartment type c

E_{i}^{k} = Binary variable to model if-then constraint 3m

A.4 Model Objective

The model objective is the goal function of the created model. This has to be similar to the goals of the planners of Company X when making the planning. This goal is described in Section 2.1 and translated into the following equation:

$$min\ \mathbb{Z} = (\ cr \sum_{k \in K} R^k) + (co \sum_{k \in K} O^k) + (cp \sum_{i \in P} \sum_{g \in G} q_{i,g} - Y_{i,g} - \sum_{k \in K} D^k_{i,g}) + (cp \sum_{i \in D} \sum_{g \in G} -q_{i,g}) + Y_{i,g} + \sum_{k \in K} D^k_{i,g})$$

The function can be split into four different components (separated by brackets), each starting with a cost parameter and then multiplied with a summation. The first component is the costs of regular hours driven. Secondly, the costs of overtime hours driven is added. The third component calculates the number of unfulfilled requests of pickup of goods. This is multiplied by the costs of not completing a pickup/delivery request. Lastly, the same costs are multiplied with the number of not delivered pallets to the delivery locations.

A.5 Model Constraints

The constraints of the model are separable in different groups: general PDP constraints; compartment constraints; vehicle constraints; objective related constraints and lastly sign constraints. In this section, constraints will be listed and discussed per group.

PDP constraints
The PDP constraints are well described in the different literature source mentioned in Section 3.1.12. The majority of this group of constraints are based upon the formulation described by Sigurd et al. ((2004)).

1a	$\sum_{k\in K}\sum_{j\in A} X^k_{i,j} \leq 1$	$\forall i \in L$
1ba	$\sum_{j\in A} X^k_{0,j} = 1$	$\forall k \in K$
1bb	$\sum_{i\in A} X^k_{i,2n} = 1$	$\forall k \in K$
1c	$\sum_{i\in A} X^k_{i,j} - \sum_{i\in A} X^k_{j,i} = 0$	$\forall j \in L,\ \forall k \in K$
1d	$B^k_i + t_{i,j} + s_i - M(1 - X^k_{i,j}) \leq B^k_j$	$\forall k \in K,\ (i,j) \in A$
1e	$e_i \leq B^k_i \leq l_i$	$\forall k \in K,\ i \in A$
1f	$X^k_{i,i} = 0$	$\forall k \in K,\ i \in A$
1g	$B^k_i + t_{i,i+n} + s_i \leq B^k_{i+n}$	$\forall k \in K,\ i \in P$
1h	$\sum_{j\in A} X^k_{i,j} - \sum_{j\in A} X^k_{n+i,j} = 0$	$\forall k \in K,\ i \in P$

1a Constraint which ensures that every location is at most served once

1b Constraint which ensures that every vehicle starts/ends at depot

1c Flow conservation constraint

1d Time flow constraint

1e Time window constraint

1f Self-travel prevention constraint

1g Pickup before delivery constraint

1h Pair pickup and delivery to the same vehicle constraint

Compartment constraints
The compartment constraints have to guide the process of the vehicle being loaded with the right quantities within the right compartment. Those compartments have to be a multiple of 3 pallet spots in the truck. Besides, the unspecified cargo must be adequately assigned to either the right compartment. All these factors together make the following group of 9 constraints.

2a	$Q^k_{2n,g} = Q^k_{0,g} = 0$	$\forall k \in K,\ g \in G$
2ba	$Q^k_{i,0} + L^{k,0}_i \leq C^{k,0}$	$\forall k \in K,\ i \in A$
2bb	$Q^k_{i,1} + L^{k,1}_i \leq C^{k,1}$	$\forall k \in K,\ i \in A$
2c	$\sum_{c\in C} C^{k,c} \leq c^k$	$\forall k \in K$
2da	$Q^k_{i,0} + Y_{i,0} - U^k_i q_{i,0} - M(1 - X^k_{i,j}) \leq Q^k_{j,0}$	$\forall k \in K,\ (i,j) \in A$
2db	$Q^k_{i,1} + Y_{i,1} - M(1 - X^k_{i,j}) \leq Q^k_{j,1}$	$\forall k \in K,\ (i,j) \in A$
2dc	$Q^k_{i,2} + Y_{i,2} - M(1 - X^k_{i,j}) \leq Q^k_{j,2}$	$\forall k \in K,\ (i,j) \in A$

2e	$C^{c,k} = \sum_{s \in S} Z^{s,c,k} w^s$	$\forall k \in K,\ c \in C$
2f	$\sum_{s \in S} Z^{s,c,k} = 1$	$\forall k \in K,\ c \in C$
2g	$\sum_{c \in C} L_i^{k,c} \geq Q_{i,2}^k$	$\forall k \in K,\ i \in A$
2ha	$M \sum_{k \in K} \sum_{i \in A} X_{i,j}^k \geq Y_{j,g}$	$\forall j \in P,\ g \in G$
2hb	$M \sum_{k \in K} \sum_{i \in A} X_{i,j}^k \geq -Y_{j,g}$	$\forall j \in D,\ g \in G$
2i	$Y_{ig} + Y_{i+ng} = 0$	$\forall i \in P,\ g \in G$

2a Empty vehicle starts at/returns to the depot

2b Compartment load capacity restriction constraint

2c Total compartment capacity constraint

2d Capacity flow constraint

2e Fixed compartment size constraint

2f 1 size of compartment constraint

2g Unspecified Load constraint

2h Restrict deliveries for unvisited locations constraint

2i Paired delivery quantities constraint

Vehicle constraints

The vehicle constraint group is the largest group of constraints, mostly focused on trailer and cool unit modelling. Trailers are modelled as a separate load space within the truck, therefore not connected to the constraints of 2d, because trailer load is vehicle dependent since not every vehicle can have a trailer, or every location is suitable for a trailer. Cool units reduce the load of the normal load of the truck when these are used for certain demand quantities. Load within trailers and cool unit has to be restricted to size and temperature environment. All these factors together creating a group of 15 constraints.

3a	$U_i^k \leq cq_i$	$\forall k \in K,\ i \in A$
3b	$\sum_{i \in A} cv_i U_i^k \leq 2$	$\forall k \in K$
3c	$U_i^k \leq ca^k$	$\forall k \in K,\ i \in A$
3d	$U_i^k = U_{i+n}^k$	$\forall k \in K,\ i \in P$
3e	$W_i^k \leq tl_i$	$\forall k \in K,\ i \in A$
3fa	$\sum_{g \in g} D_{ig}^k \leq MW_i^k$	$\forall k \in K,\ i \in P$
3fb	$-\sum_{g \in g} D_{ig}^k \leq MW_i^k$	$\forall k \in K,\ i \in D$
3g	$\sum_{i \in A} W_i^k \leq 2$	$\forall k \in K$
3h	$W_i^k = W_{i+n}^k$	$\forall k \in K,\ i \in P$
3ia	$M \sum_{i \in A} X_{ij}^k \geq D_{jg}^k$	$\forall k \in K,\ j \in P,\ g \in G$
3ib	$M \sum_{i \in A} X_{ij}^k \geq -D_{jg}^k$	$\forall k \in K,\ j \in D,\ g \in G$
3ja	$\sum_{i \in P} \sum_{g \in G} D_{ig}^k \leq ts^k$	$\forall k \in K$
3jb	$-\sum_{i \in D} \sum_{g \in G} D_{ig}^k \leq ts^k$	$\forall k \in K$
3k	$\sum_{cinC} T^{kc} \leq 1$	$\forall k \in K$

3la	$\sum_{i \in P} D^k_{i0} \leq MT^{k1}$	$\forall k \in K$
3lb	$\sum_{i \in D} D^k_{i0} \leq MT^{k1}$	$\forall k \in K$
3lc	$\sum_{i \in P} D^k_{i1} \leq MT^{k0}$	$\forall k \in K$
3ld	$\sum_{i \in D} D^k_{i1} \leq MT^{k0}$	$\forall k \in K$
3ma	$W^k_i \leq ME^k_i$	$\forall k \in K,\ i \in P$
3mb	$-B^k_i + B^k_{i+n} - t_{ii+n} - s_i \leq M(1 - E^k_i)$	$\forall k \in K,\ i \in P$
3mb	$B^k_i - B^k_{i+n} + t_{ii+n} + s_i \leq M(1 - E^k_i)$	$\forall k \in K,\ i \in P$
3na	$\sum_{k \in K} D^k_{ig} + Y_{ig} \leq q_{ig}$	$\forall i \in P,\ g \in G$
3nb	$\sum_{k \in K} D^k_{ig} + Y_{ig} \geq q_{ig}$	$\forall i \in D,\ g \in G$
3o	$lr^k_j - \sum_{i \in A} X^k_{ij} \geq 0$	$\forall k \in K,\ j \in L$

3a Cooled demand suitability constraint

3b Max load of cool unit constraint (2, since pickup and delivery both occupy space)

3c Cool Unit presence constraint

3d Pair cool unit usage constraint

3e Trailer suitability constraint

3f Trailer demand serving constraint

3g Limited customers served from trailer constraint

3h Paired Trailer serving constraint

3i Visit to serve from trailer constraint

3j Trailer capacity constraint

3k Trailer temperature constraint

3l Trailer items matches temperature constraint

3m Trailer first emptied constraint

3n Trailer or compartment allocation constraint

3o Limited locations constraint

Objective related constraints
This group of constraints contains the definitions of the variables determining the objective value of the problem.

4a	$B^k_{2n} - B^k_0 \leq O^k + R^k$	$\forall k \in K$
4b	$R^k \leq ol$	$\forall k \in K$
4c	$O^k + R^k \leq mdt$	$\forall k \in K$

4a Total drive time constraint

4b Regular hours constraint

4c Max drive time constraint

Sign constraints
The last group of constraints in the that bound the values of the variables and defines their type and size. Their type can be continuous, binary or integer. The size can be completely bounded or unbounded, or partly bounded with either an upper bound or a lower bound. When variables have the same type and size, they are mentioned in the same line.

5a	$X_{ij}^{k} \in \{0,1\}$	$\forall k \in K,\ (i,j) \in A$
5b	$Z^{sck} \in \{0,1\}$	$\forall k \in K,\ s \in S,\ c \in C$
5c	$R^{k},\ O^{k} \geq 0$	$\forall k \in K$
5d	$C^{kc} \geq 0$	$\forall k \in K,\ c \in C$
5e	$B_{i}^{k} \geq 0$	$\forall k \in K,\ i \in A$
5f	$Q_{ig}^{k} \geq 0\ integer$	$\forall k \in K,\ i \in A,\ g \in G$
5g	$L_{i}^{kc} \geq 0\ integer$	$\forall k \in K,\ c \in C,\ i \in A$
5h	$0 \leq Y_{ig} \leq q_{ig}\ integer$	$\forall i \in A,\ g \in G$
5i	$0 \leq D_{ig}^{k} \leq q_{ig}\ integer$	$\forall k \in K,\ i \in A,\ g \in G$
5j	$U_{i}^{k},\ W_{i}^{k},\ W_{i}^{k} \in \{0,1\}$	$\forall k \in K,\ i \in A$
5k	$T^{kc} \in \{0,1\}$	$\forall k \in K,\ c \in C$

B Constructive Algorithm

Algorithm 1 Constructive Route Algorithm

Input: Orders, Locations, Vehicles, Standard Routes
Output: Routes & **Unfulfilled Orders**

Create **Unplanned Orders** which are **Orders** of **Locations** not in **Standard Routes**
for each Vehicle $\in$ **Vehicles do**
  Find optimal configuration for **Orders** of **Standard Routes** of **Vehicle**
  if if **Order** not in optimal configuration **then**
    Remove **Order** from **Vehicle**
    Add **Order** to **Unplanned Orders**
  end if
end for

Divide **Unplanned Orders** into lists for small and 'normal' **Locations**
if Σ capacity of small location **Vehicles** > 0 **then**
  if Σ load **Orders** small locations $> \Sigma$ Capacity small location **Vehicle then**
    Calculate weighted centre points of small order **Locations** in **Vehicle**
    Calculate Distances to weighted centre points for unplanned small orders
    while do Σ load **Orders** small locations $> \Sigma$ Capacity small location **Vehicle**
      Select **Order** with largest distance to weighted centre point
      Add **Order** to **Unfulfilled Orders**
      Remove **Order** from **Orders** small locations list
    end while
  end if

  Sort **Orders** small locations descending Load
  for each Order small locations **do**
    for each Vehicle $\in$ **Small Vehicles do**
      Calculate insertion costs **Order** into **Vehicle**
    end for
    Insert **Order** to **Route** with the lowest insertion costs
  end for

else
  Add list **Orders** small locations to **Unfulfilled Orders**
end if

Sort (non-small) unplanned **Order** list on descending load
for each unplanned **Order** $\in$ **Orders do**
  Determine **Vehicles** which can fit this **Order**
  if Vehicles to fit **Order** > 0 **then**
    for each Vehicle $\in$ **Vehicles do**
      Calculate insertion costs **Order** into **Vehicle**
    end for
    Insert **Order** to **Route** with lowest insertion costs
    Remove **Order** from Unplanned **Order** list
  else
    Add **Order** to **Unfulfilled Orders**
    Remove **Order** from Unplanned **Order** list
  end if
end for

Return **Routes** & **Unfulfilled Orders**

C Optimization Algorithm

Algorithm 2 Optimization Algorithm

Input: Orders, Locations, Vehicles, & **Standard Routes,** & **Initial ALNS Parameters,** & **RunTime,** & **UpdateTime**
Output: BestSolution

```
CurrentSolution = ConstructiveHeuristic(Orders, Locations, Vehicles, Standard Routes)
BestSolution = CurrentSolution
AdaptiveParameters = Initial ALNS Parameters
StartTime, CurTime, LastUpdateTime = Time()

while CurTime - StartTime < RunTime do

    NewSolution = CurrentSolution
    DegreeOfDestruction = RetrieveDegreeOfDestruction(Initial ALNS Parameters,
    CurTime, len(Orders))
    DestoryMethod = SelectDestroyMethod(AdaptiveParameters)
    RepairMethod = SelectRepairMethod(AdaptiveParameters)
    NewSolution = Destory(CurrentSolution,DestoryMethod,DegreeOfDestruction)
    NewSolution = Repair(CurrentSolution,RepairMethod)

    if NewSolution < CurrentSolution then
        if NewSolution < BestSolution then
            BestSolution = NewSolution
            Update SuccesRatio
        else
            Update SuccesRatio
        end if
        CurrentSolution = NewSolution
    else
        AcceptanceValue = (NewSolution - BestSolution) / NewSolution
        Threshold = 1 - ((CurTime - StartTime) / RunTime)
        if AcceptanceValue < Threshold then
            CurrentSolution = NewSolution
            Update SuccesRatio
        else
            Update SuccesRatio
        end if
    end if

    Update Usages
    Update CurTime

    if CurTime - LastUpdateTime > UpdateTime then
        Update AdaptiveParameters
        Update LastUpdateTime
    end if

end while
```

C.1 Random Removal Heuristic

Algorithm 3 Random Removal Heuristic

Input: ProblemInstance, DegreeOfDestruction
Output: ProblemInstance

Create List of possible destructions per vehicle
Calculate maximal number of destructions
for DestoryIndex in range(0, min(**Degree Of Destruction**,MaxNumDestructions)) **do**
    Get random vehicle to pick from
    Get random order pair (pickup and delivery) from destruction list of random vehicle
    Remove orders from vehicle
    Add order to Unfulfilled Orders
    Remove order from destruction list
end for

C.2 Greedy Removal Heuristic

Algorithm 4 Random Removal Heuristic

Input: ProblemInstance, DegreeOfDestruction
Output: ProblemInstance

Calculate maximal number of destructions
Create destruction cost list

for DestoryIndex in range(0, min(**Degree Of Destruction**,MaxNumDestructions)) **do**
    Get the lowest cost order pair
    Remove orders from vehicle
    Add order to Unfulfilled Orders
    Update destruction cost list
end for

C.3 Greedy Repair Heuristic

Algorithm 5 Random Removal Heuristic

Input: ProblemInstance
Output: ProblemInstance

Create additional cost matrix
Find cheapest insertion

while Insertion is profitable **do**
    Insert Order into vehicle
    Update additional cost matrix
    Find new cheapest insertion
end while

C.4 Regret Repair Heuristic

Algorithm 6 Random Removal Heuristic

Input: ProblemInstance
Output: ProblemInstance

Create additional cost matrix
Calculate Regret values for each unfulfilled order
Find insertion with most regret

while Insertion is profitable **do**
    Insert Order into vehicle
    Update additional cost matrix
    Recalculate regret values
    Find new insertion with most regret
end while

D Assumptions, Limitations & Simplifications Explanations

In this appendix, some assumptions, limitations or simplifications will be further highlighted.

D.1 Number 13

Calculating the insertion costs of an order into a vehicle is one of the largest components of the constructive heuristic, as well as for the repair heuristics. The insertion costs are dependent on the order position it will get in the vehicle. This means that for a problem the size of Company X we have to calculate for each 36 vehicles with on average 5 orders per vehicle, 36*(5+1) = 216 possibilities per new inserted order. This is a time-consuming process with is not realistic and beneficial. Namely, most of the vehicles are full or cannot store a large part of the order. On top of that, customer are not really pleased if only a fraction of the order is served. Therefore, we do not calculate the insertion costs if less than 25% of the order fit the vehicle. For example, if the vehicle is full, nothing will fit, and we will not calculate the insertion costs. If the remaining capacity is 2, we will calculate insertion costs for orders up to 8 pallets.

D.2 Number 19

This assumption causes that infeasible insertions are not allowed in vehicles. This choice is twofold, it results in that all solutions created are feasible. The disadvantages of this choice is the reduction in the neighbourhood size of the solution space and consequently makes it more difficult to find good solution neighbourhoods. To prevent searching constantly in infeasible neighbourhoods, a (increasing) penalty costs can be assigned to force feasibility towards the end of the procedure. In this research, we decide to only allow feasible insertions, instead of working with penalty costs for infeasibility. The main reason to use only feasible solutions is the problem context of Company X. Due to the complexity, the solution space is enormous (See section 4.2.2). Consequently, finding feasible solutions with reasonable solution values is already time-consuming. When allowing infeasible solutions, there is a risk of not finding a feasible solution in a given time to the improvement algorithm. This is not desirable for Company X in a procedure to create schedules. An additional reason for not using penalty values is that the value of the penality is difficult to determine, and therefore requiring additional experimentation, for example, on variable and fixed values.

D.3 Number 36

The simplification that colli from the order data can be not only be merged into new pallets, but also into existing pallets. Firstly, we check how many standard and 15-25 degree pallets are present in the order. For each of these pallets, we can add (if they exist) 10 colli of both types on top of these pallets. In this way, the number of pallets is reduced in practise. In reality, this number of 10 can be higher or lower or even 0, but 10 is agreed upon with the planners of Company X the default value they work with. One example to illustrate the aggregation is can be the following order. An order with 3 standard pallets and 50 colli, 4 15-25 pallets and 80 colli for a hospital (40 colli makes one pallet for this customer type). There are 4 pallets to merge 40 colli of the total 130 colli upon, this makes that we have to make new pallets for the remaining 90 colli (Note: we do not merge colli upon a standard pallet). These 90 colli construct into 3 new pallets, since we have to round the number of pallets upwards. In total, we have 3 standard pallets and 7 15-25 pallets from the initial order.

D.4 Number 41

Order prioritization is one of the factors which can be easily account for in manual plannings but not in automated plannings. It is technically possible to add an attribute to an order and assign a cost factor for not serving this order. However, with the given data, it is not possible to assign priorities to orders. Therefore, we did not take this into account in the model. However, the model does indirectly prioritize larger orders over small order. This is since the costs for not serving is per pallet, instead of per order. This makes serving 1 order of 8 pallets more indirectly (less distance and service time) more beneficial than 8 orders with 1 pallet. With this effect, we do not see many benefits in including this effect in the model.

D.5 Number 43

Multiple trips is common practise at Company X. However, there are two reason why we do not model them in both the constructive algorithm and the improvement algorithm. The first one being the fact that current decision of second trips are planned in the continuation planning (see Section 2.1.6) whilst our focus is on the distribution planning (see Section 2.1.4). Taking second trips into consideration would broader our scope by including another operation planning. The second reason is that there is no data present on continuation and direct orders. These are not orders planned in second routes, but influences which vehicles are suitable for second trips. Resulting in reduced practical usability of these plannings. Therefore, we do not consider second trips in the current algorithms. However, the used structure enables relative simple changes to do that (see Section 6.1).

E Experimentation Literature

Table 23: Literature Table

Source	**Update/Stop Criterion**	**Degree Of Destruction**	**Success Rate** $[\sigma_1, \sigma_2, \sigma_3, \sigma_4]$	**Roulette Wheel Parameter**
((Majidi et al., 2018))	800/50.000 iterations	10% of customers, decrease during procedure	[10,0,1,?]	0.1
((Majidi et al., 2019))	100/5.000 or 10.000 iterations	values between 10% and 25% of customers, increases when solutions are rejected	[0.7,0.1,0.2,?]	0.5
((Sacramento et al., 2019))	-/time, stopped after 1.000 no improving iterations	15% of customers	[33,9,13,0]	0.9
((Eshtehadi et al., 2020))	100/25.000	Values between $log_{1.4}$ and log_{10} of customers	[10,5,1,?]	0.1
((Chowmalia and Sukto, 2021))	-/200.000	30% of customers	-	-
((Emeç et al., 2016))	100/25.000, stopped after 4.000 no improving iterations	randomly between min(10% of customers, 30 customers) and min(40% of customers, 60 customers)	[20,16,13,?]	0.1
((Liu et al., 2019))	Simulating annealing principle, unclear values	values between 15% and 30% of customers	[33,9,13,?]	0.1
((Li et al., 2020))	100-≈200/20.000	-	Classified according to relative distance between new and current best solution: $>$ 10% [90,30,5,?]; $>$ 1% [80,25,5,?]; $>$ 0.1% [70,20,5,?]; $>$ 0.01% [60,15,5,?]; $<$ 0.01% [50,10,5,?]	increasing form 0.001 to 0.999 during the procedure

F Schedule Comparison Results

This appendix provides a table with all performance measures collected from the three schedules used to compare manual scheduling with the constructive and improvement algorithms designed

Table 24: Schedule Comparison Results

Attribute Name	**Manual**	**Constructive**	**Improvement**
Total Costs	€49.903	€42.429	€36.894
Regular Time Costs	€7.804	€8.205	€8.198
Overtime Costs	€590	€716	€370
Kilometre Costs	€4.709	€4.708	€4.126
Unserved Orders Missed Pallet Costs	€14.200	€18.000	€21.400
Served Orders Missed Pallet Costs	€22.600	€10.800	€2.800
Work Costs	€13.103	€13.629	€12.694
Missed Pallet Costs	€36.800	€28.800	€24.200
Total Missed Pallets	184	144	121
CoolUnit Pallets	33	28	38
Empty Capacity	62	17	4
Empty Capacity %	7.3%	2.1%	0.5%
Total Worktime (h)	256	271	264
Total Drivetime (h)	171	201	197
Average Unused Capacity	7.3%	1.5%	0.3%
Average Overplanned Capacity	15.1%	7.6%	2.0%
Average Number of Stops	6.8	5.2	4.8
Number of Standard Orders	104	125	17

G Random Repair

Algorithm 7 Random Repair Heuristic

Input: ProblemInstance
Output: ProblemInstance

```
Create list of vehicles with capacity to insert into
for Vehicle in VehicleInsertionList do
    Create List of feasible orders to insert in Vehicle
end for
Combine lists of insertion orders per vehicle into one list
while len(InsertionOrderList) > 0 do
    Choose random order to insert into vehicle
    Apply insertion and update problem
    Remove insertions from list of this order
    if Vehicle capacity is complete used then
        remove insertions into this vehicle from list
    end if
end while
```

H Statistical Comparison

This appendix presents the statistical values and calculations for the findings of Section 5.9.2. Section H.1 provides the calculations for the confidence intervals and the expected number of samples for non-overlapping intervals. Section H.2 elaborates upon the outcomes of the two sample t-test.

H.1 Confidence Intervals

For the calculation of the confidence interval, we retrieved the sample mean and standard deviations from both samples, as shown in Section 5.9.2. The individual experiment results are in Table 25. With the mean μ of the samples, we can calculate the maximum width the confidence can have to not overlap, confidence interval half width (CIHW). These widths have to be sufficient small, and this can only be done reliably reduced by increasing the sample size. Since the formula for the CIHW is: $CIHW = T.value(\alpha = 0.05, n) * \frac{\sigma}{\sqrt{n}}$. To achieve non-overlapping intervals, the CIHW cannot be larger than half of the difference between the two sample means. The n^* can be found by solving the following formula:

$$n^* = \frac{T.value(\alpha = 0.05, n^*)}{\frac{maxCIHW}{\sigma}^2}$$

By matching the n^* on both sides of the equation, we can find the total required number of samples of 130.

Table 25: Robustness Experiment Result Summary

(a) Lowest Costs Setting Results

Experiment Seed Value	**Total Costs**
100	€38.016
200	€38.208
300	€37.345
400	€37.495
500	€38.068
600	€38.686
700	€38.674
800	€38.763
900	€37.626
1000	€39.010
1100	€38.876
1200	€38.276
1300	€37.800
1400	€38.206
1500	€37.404
1600	€38.276
1700	€37.587
1800	€37.691
1900	€38.969
2000	€39.321
2100	€37.328
2200	€37.956
2300	€37.458
2400	€39.097
2500	€37.432

(b) Best Grid Search Setting Results

Experiment Seed Value	**Total Costs**
150	€39.275
250	€37.661
350	€38.480
450	€37.890
550	€36.861
650	€38.532
750	€38.437
850	€38.412
950	€40.807
1050	€38.303
1150	€38.180
1250	€38.612
1350	€38.301
1450	€38.845
1550	€38.141
1650	€38.656
1750	€38.007
1850	€38.489
1950	€38.568
2050	€39.443
2150	€37.681
2250	€37.908
2350	€38.133
2450	€38.399
2550	€37.530

H.2 Two Sample t-test

For performing the two sample t-test, we used a function with the data analysis function of Excel. The t-test: Two-Sample Assuming Unequal Variances generated the information of Table 26.

Table 26: t-test: Two-Sample test results

	LowCost	**GridSearch**
Mean	€38.143	€38.382
Variance	€394.560	€544.172
Observations	25	25
Hypothesized Mean Difference	0	
df	47	
t Stat	-1.23	
$P(T \leq t)$ one-tail	0.11	
t Critical one-tail	1.68	
$P(T \leq t)$ two-tail	0.22	
t Critical two-tail	2.01	

I Mappings

In this appendix, you find both the export and import mapping to call upon the solver using JSON.

I.1 Export Mapping

In the figure below, the complete Import Mapping is visualized.

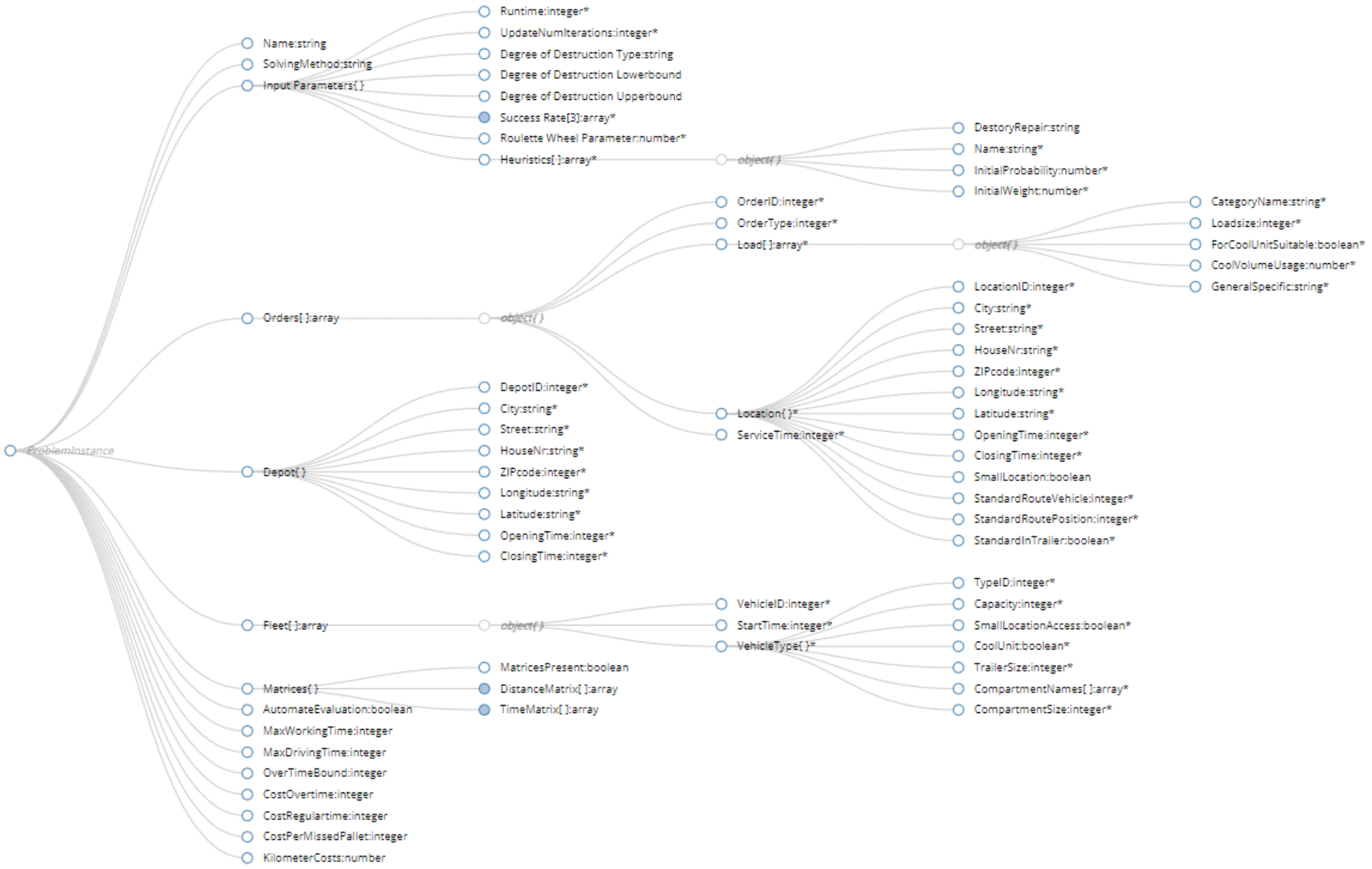


Figure 39: Visualization of the import mapping to problem to the implementation

I.2 Import Mapping

In the Figure below the complete Import Mapping is visualized.

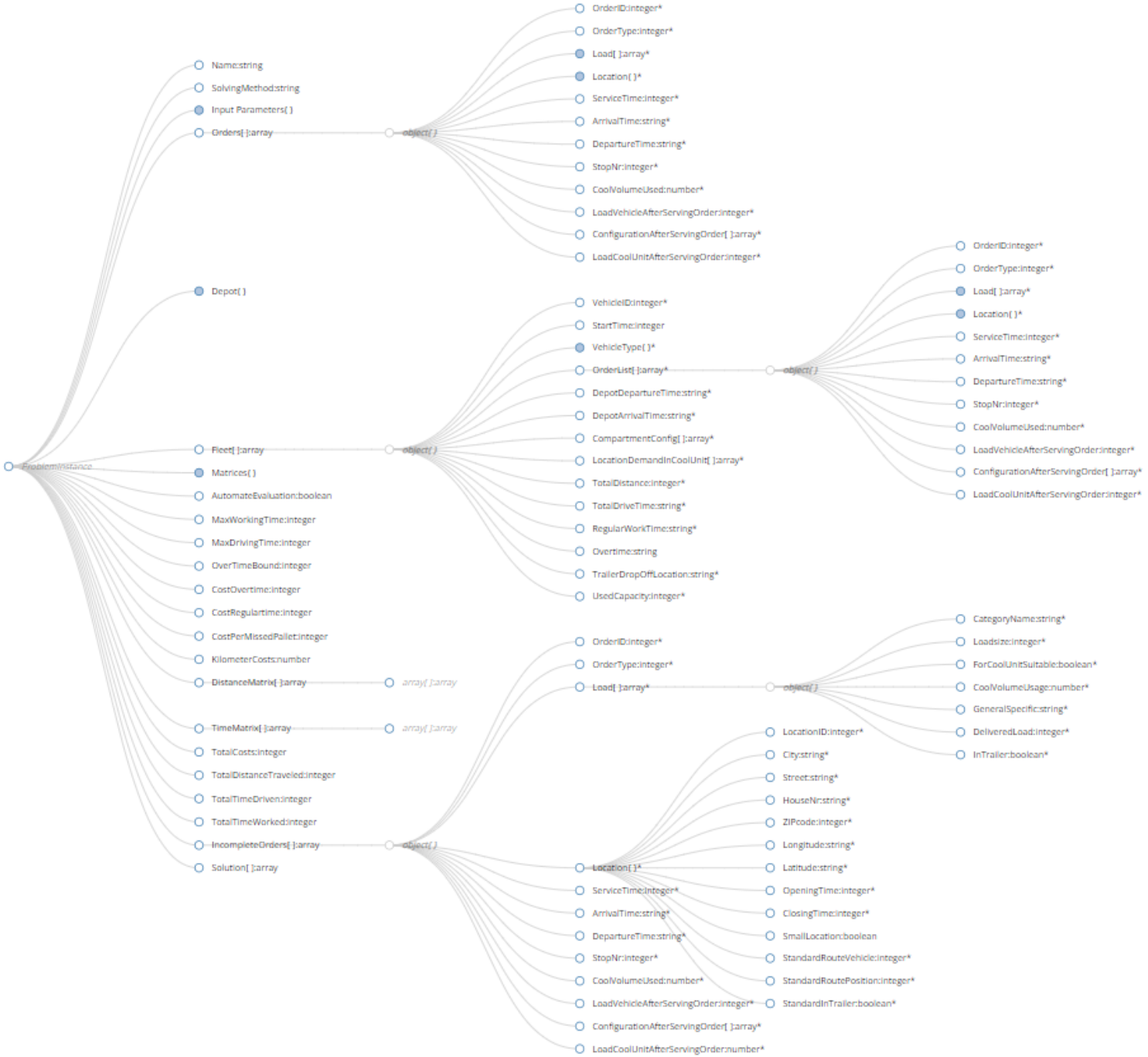


Figure 40: Visualization of the import mapping used to map the results back to the application

J Attribute Descriptions

This appendix provides descriptions of each attribute required to generate a solution from the designed structure. The attributes are divided on the presence in the export or import mappings. Within these divisions, the descriptions are grouped per object. The import attributes do not contain the export attributes already described.

J.1 Export Attribute Descriptions

Table 27: Description of all attributes necessary to do a complete export from an application to the structure

(a) Problem Instance Attributes

Attribute	**Description**	**Data Type**
Name	Name of the problem to identify the problem	string
SolvingMethod	Solving method to use in the solving	string enum: "Constructive", "Improvement"
InputParameters	Object for input parameters to run the improvement algorithm	object
Depot	Object to identify the depot	object
Orders	List of orders to schedule in the problem	array (Order objects)
Fleet	List of vehicles to schedule in the problem	array (Vehicle objects)
Matrices	Object to identify whether we have to create distance and time matrices or not	object
Automate Evaluation	Boolean to indicate whether an excel evaluation has to be made	boolean
MaxWorkingTime	Max allowed working time for drivers in seconds	integer
MaxDrivingTime	Max allowed driving time for drivers in seconds	integer
OvertimeBound	Time in seconds after which drivers work in overtime	integer
CostOvertime	Costs for working in overtime, euros per hour	integer
CostRegulartime	Costs for working in regular time, euros per hour	integer
Cost Per Missed Pallet	Costs for not delivering 1 pallet to a customer in euros	integer
KilometerCosts	Costs for driving a kilometer with a vehicle	number

(b) Order Attributes

Attribute	**Description**	**Data Type**	**Note:**
OrderID	Identification number for an order	integer	Pickup and Delivery order need same OrderID
OrderType	Number to define order type: 1 = pickup, 2 = delivery	integer enum: 1,2	-
Load	List of the Load for this order	array (Load objects)	-
Location	Location to which an order has to be served	object	-
ServiceTime	Time spend to serve the order in seconds	integer	-

(c) Location Attributes

Attribute	Description	Data Type
LocationID	Identification number for a location	integer
City	City of a location	string
Street	Street of a location	string
HouseNr	House number of a location	string
ZIPcode	ZIP code of a location	string
Longitude	Longitude of a location	string
Latitude	Latitude of a location	string
OpeningTime	Opening time of a location in seconds	integer
ClosingTime	Closing time of a location in seconds	integer
SmallLocation	Boolean whether a location is only suitable for small vehicles	boolean
Standard Route Vehicle	VehicleID of the vehicle if the location has a standard route in this vehicle, -1 otherwise	integer
Standard Route Position	Position within the standard route of a vehicle, 0 otherwise	integer
Standard In Trailer	Boolean to define if the location is standard located in a trailer of the StandardRouteVehicle	boolean

(d) Load Attributes

Attribute	Description	Data Type
CategoryName	Name of the category for this load object	string
Loadsize	Size of the load of this load object	integer
ForCoolUnitSuitable	Load suitable for a cool unit	boolean
CoolVolumeUsage	Usage percentage of the load if loaded in a cool unit	number
GeneralSpecific	Enumeration to define whether this load has to be loaded into a specific compartment or it can be load into any compartment	string enum: "General","Specific"

(e) Vehicle Attributes

Attribute	Description	Data Type
VehicleID	Identification number for an vehicle	integer
VehicleType	Type object of the vehicle	object
StartTime	Desired time for the vehicle to start its route	integer

(f) Vehicle Type Attributes

Attribute	Description	Data Type
TypeID	Identification number for the vehicle type	integer
Capacity	Capacity in pallets for the type	integer
SmallLocationAccess	Boolean if a vehicle can access small locations	boolean
CoolUnit	Boolean if a vehicle has a cool unit	boolean
TrailerSize	Capacity of the trailer, 0 if no trailer is present	integer
CompartmentNames	List of names for each compartment	array (of strings)
CompartmentSize	Integer to describe how many load creates one comparment for comparment seperation	integer

(g) Depot Attributes

Attribute	**Description**	**Data Type**
DepotID	Identification number of the depot	integer
City	City of the depot	string
Street	Street of the depot	string
HouseNr	House number of the depot	string
ZIPcode	ZIP code of the depot	string
Longitude	Longitude of the depot	string
Latitude	Latitude of the depot	string
OpeningTime	Opening time of the depot in seconds	integer
ClosingTime	Closing time of the depot in seconds	integer

(h) Matrices Attributes

Attribute	**Description**	**Data Type**	**Note:**
MatricesPresent	Boolean to state whether this object has distance and time matrices	boolean	-
DistanceMatrix	Matrix to define distances between locations and the depot	array	Not required when Matrices Present is false
TimeMatrix	Matrix to define travel times between locations and the depot	array	Not required when Matrices Present is false

(i) Input Parameters Attributes

Attribute	**Description**	**Data Type**	**Note:**
RunTime	Number of seconds the improvement heuristic is allowed to run	integer	-
Update Num Iterations	Number of iterations allowed between updating the ALNS parameters	integer	-
Degree Of Destruction Type	Input parameter to state which type of destruction will be used	string	One of the following: "Flat Number", "Flat Percentage", "Flat Number Decreasing", "Flat Percentage Decreasing", "Random Interval Flat", "Random Interval Percentage", "Increase Rejection Interval Flat", "Increase Rejection Interval Percentage", "Mixed Random Interval"
Degree Of Destruction Lowerbound	Input for the lower bound to determine how much of the solution will be destroyed	see Note:	number required for all except "Mixed Random Interval" which should be string
Degree Of Destruction Upperbound	Input for the upper bound to determine how much of the solution will be destroyed	see Note:	number required for all except "Mixed Random Interval" which should be string
Success Rate	Input parameter how successful found solution are valued	array	list of 3 values, first global success factor, second current success factor, third worse but accepted value
Roulette Wheel Parameter	Input parameter to determine how much the weights and probabilities are adjusted per update	number	-
Heuristics	List of Heuristic Objects to run the improvement heuristic	array(Heuristic objects)	-

(j) Heuristic Attributes

Attribute	Description	Data Type
DestroyRepair	String to define whether this is a destoy or repair heuristic	string enum: "Destroy","Repair"
Name	The name of the heuristic	string
InitialProbability	Initial probability to choose this heuristic in the procedure	number
InitialWeight	Initial weight of the heuristic	number

J.2 Import Attribute Descriptions

Table 28: Description of all attributes necessary to do a complete import from the structure to an application

(a) Problem Instance Attributes

Attribute	Description	Data Type
Depot	Object to identify the depot	object
Orders	List of orders to schedule in the problem	array (Order objects)
Fleet	List of vehicles to schedule in the problem	array (Vehicle objects)
DistanceMatrix	Matrix used to determine distances between locations	array (array (number))
TimeMatrix	Matrix used to determine travel times between locations	array (array (number))
TotalCosts	Total Costs of the best solution found	integer
TotalDistance Traveled	Total distance traveled by all vehicles	integer
TotalTime Driven	Total time driven by all drivers	string
TotalTime Worked	Total time worked by all drivers	string
IncompleteOrders	List of Orders which are not sered at all	array (Order objects)
Solution	List which contains easily evaluatable data of the problem solution	array

(b) Order Attributes

Attribute	Description	Data Type
ArrivalTime	Time at which the vehicle arrives the order location	string
DepartureTime	Time at which the vehicle departs the order location	string
StopNr	Number to indicate which stop the order is within the vehicle route	integer
CoolVolumeUsed	Volume of the cool unit used by the load of the order	number
Load Vehicle AfterServing	Load of the vehicle after serving the order	integer
Configuration AfterServing	Configuration of the vehicle after serving the order	array (integer)
Load CoolUnit AfterServing	Load of the cool unit after serving the order	number

(c) Load Attributes

Attribute	Description	Data Type
DeliveredLoad	Load delivered to the customer	integer
InTrailer	Boolean to define if the load is served from trailer or not	boolean

(d) Vehicle Attributes

Attribute	Description	Data Type
OrderList	List of orders which are served by the vehicle	array (Order objects)
Depot DepartureTime	Time at which the vehicle departs the depot	string
Depot ArrivalTime	Time at which the vehicle arrives at the depot	string
Compartment Config	(Largest) Configuration of the vehicle	array (integer)
Location Demand In CoolUnit	Array of cities which demands are located in the cool unit	array (string)
TotalDistance	Distance traveled by the vehicle in kilometers	integer
TotalDrivetime	Time driven by the vehicle	string
RegularWorktime	Time worked by the vehicle within the regular time bound	string
Overtime	Time worked in overtime by the vehicle	string
Trailer Drop Off Location	City in which the vehicle drops the trailer	string
UsedCapacity	Max capacity used by the vehicle	integer

(e) Order Attributes

Attribute	Description	Data Type
Earliest Departure	Earliest departure of a vehicle from the depot	string
Latest Departure	Latest departure of a vehicle from the depot	string
Earliest Arrival	Earliest arrival of a vehicle from the depot	string
Latest Arrival	Latest arrival of a vehicle from the depot	string