Upload! – Transportation planning and marketplace for inefficient trips

Construction of the mathematical intelligence of the 'Upload!' route planning and marketplace.

MASTER THESIS

INDUSTRIAL ENGINEERING & MANAGEMENT

UNIVERSITY OF TWENTE

Author: Romke Frits Veenstra BSc
Student M-IEM (s0204668) at University of Twente

Date: Wednesday, April 12th 2017

University of Twente

Dr. Ir. M.R.K. Mes
Dr. Ir. J.M.J. Schutten

InfoStructure

Ing. A. Oldenburger

MANAGEMENT SUMMARY

InfoStructure, a company focused on travel and information technology is planning to release a transportation planning and load sharing platform called 'Upload!'; targeted at transportation companies with a relative small amount of vehicles. Over 75% of the Dutch transportation companies have 5 employees or less, indicating that the market share of small transportation companies is large. For these companies, inefficient orders (i.e. orders that cause high transportation costs compared to the transportation fee) are difficult to relocate to other transportation companies, since existing load sharing solutions require manual monitoring and sharing of freight, wherefore smaller companies do not have the required manpower. Furthermore these companies don't have the economy of scale larger companies have.

This transportation planning and load sharing platform InfoStructure aims to create must consists of three parts, namely a) the routing problem, b) selection of inefficient orders and trips and c) a matching algorithm that seeks how to exchange inefficient orders, see Figure 1. Aim of this thesis is to create the mathematical intelligence, i.e., the algorithms needed to find a solution to the three parts of 'Upload!'.

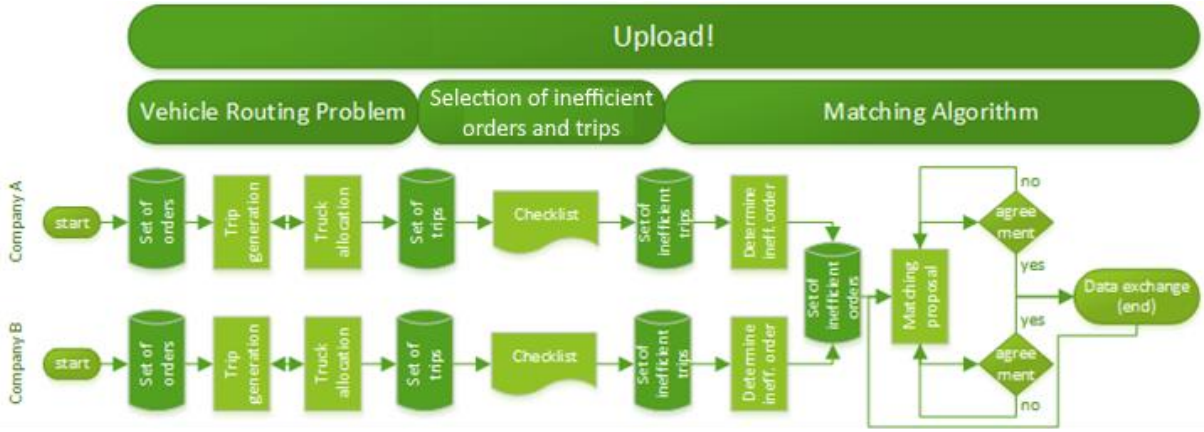


Figure 1: High level architecture of 'Upload!'

We propose the following solution methods for the three parts:

Vehicle routing problem

To solve the vehicle routing problem, we first create a framework showing which vehicle routing constraints are part of the 'Upload!' problem setting. Then we decide which constraints we actually implement, the selected constraints, and which constraints we do not use in this research but must become part of 'Upload!' in a later stage, the neglected constraints, see Figure 2. Based on these constraints we then select and implement two algorithms; a savings based algorithm to construct an initial solution and a simulated annealing algorithm to improve the solution.

For the simulated annealing we use 3 operators, a pointmove which moves a point within a trip, an ordermove which moves an order from one trip to another and an orderswap which swaps two orders between trips. We selected parameters for the goal function and the simulated annealing based on the Li Lim benchmark for VRPs with pickup and delivery and time windows. For the Li Lim benchmark with 100 customers we find results within about 5% of the optimal value after 5 minutes of runtime.

We further improved the quality of the solution by temporarily allowing time window and capacity violations at an additional cost factor. Furthermore we introduced Large Neighbourhood Search (LNS) as an additional local search operator. We found that the introduction of LNS can improve the quality of the solutions found, but the improvement comes at the cost of a long runtime.

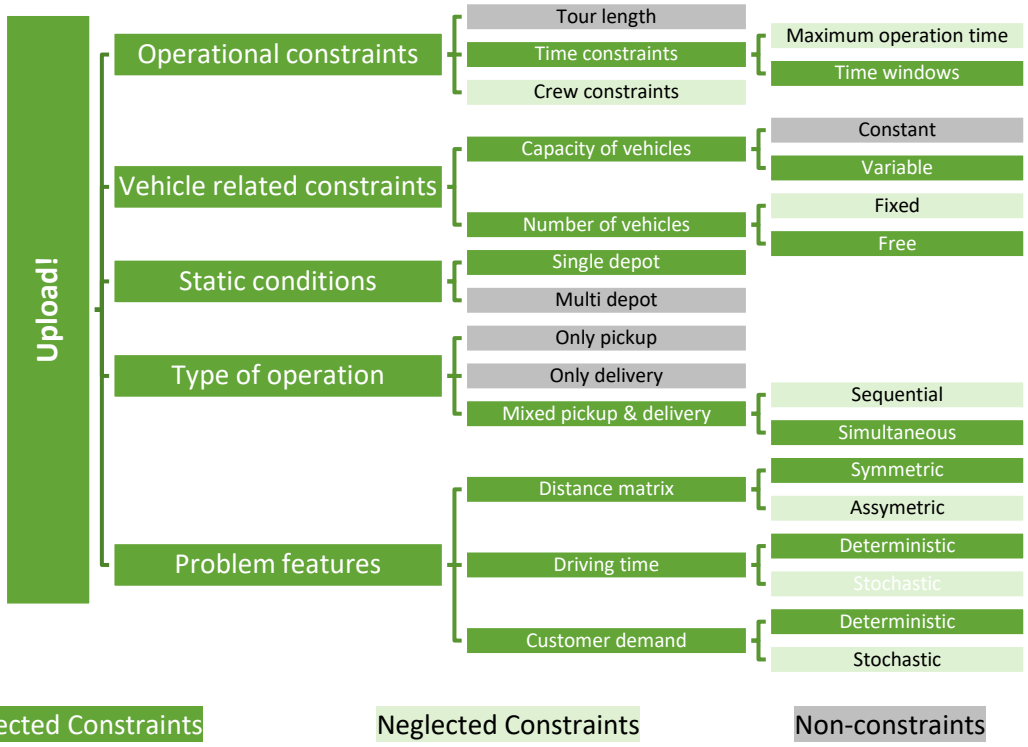


Figure 2 Framework showing the selected constraints that are going to be implemented in the proof of concept

Selection of inefficient orders and trips

For the selection of inefficient orders and trips we propose a method of assessing the quality of trips based on financial and empirical Performance Indicators (PIs). For the financial PI we compare the operation costs of the trip with the expected revenue. We also look at 4 empirical PIs, a) the number of orders per trip, b) the waiting time of a trip, c) the relative distance of the trip compared to the distance of delivering each order in a separate truck and d) the load factor. We found a clear correlation between each of these empirical PIs and the financial PI, indicating that financial performance improvement also leads to improvement of the tested empirical measures.

Then we construct a framework to determine which orders of an inefficient trip cause the inefficiency of the trip as a whole. The benchmark data however is constructed in such a way that no meaningful parameters for this framework could be found.

Matching algorithm

For the matching algorithm we propose the Vickrey-Clarke-Groves auction as an exchange method. This combinatorial auction mechanism determines the best allocation of freight amongst the participants. Characteristics include that truthful bidding is optimal for all participants and just a single bid per participants at every order combination is required. The auction method is implemented using an exact method (Dynamic Programming, DP) to determine the optimal allocation among the participants. Results show that order sharing leads to better results for all participants, even if the distance between participants increases to up to 1.5 times the service area

of the participants. Drawback of the exact implementation is that only results up to 4 participants and 13 inefficient orders can be calculated in reasonable time.

Overall we conclude that important steps are taken to implement the mathematical intelligence of 'Upload!', but that there is additional research required, mentioned in the section "Recommendations for further research", before implementation can be done.

PREFACE

This graduation thesis document in front of you marks the end of my time as a student at the University of Twente. During the last years I haven't really been a student, visited Enschede only when I had an appointment with my supervisors and didn't really felt like a student anymore. This graduation however does give me an emotional feeling, the feeling that it is "all over now". Where there is an end however, will always be a new begin, and that new begin has already begun, by the time I graduate I'm already working at ARS T&TT, the mother company of the company I graduated.

My time at InfoStructure confirmed that it was good for me to switch from a Bachelor in Chemical Engineering to a Master in Industrial Engineering & Management. The rush of excitement when after days of implementing algorithms I see the results of my work is indescribable. At the same time there were difficult periods and quite a few times I was unsure whether I would be able to finish this project. Overall however, the strength to continue was stronger than all struggles along the road.

I want to thank the people who made it possible for me to finish this thesis. I would like to thank all of them for their enthusiasm, their trust and their patience. In particular I want to thank my supervisors for their guidance and advice. Martijn Mes, who started with the warning that "during the meeting we only discuss what needs improvement", yet still encouraged me to continue my work and showed me he believed in this result, Marco Schutten, who checked every dot and index of my formula's and so improved the quality of my work and André Oldenburger, who was able to get the most out of me. I also would like to thank my girlfriend Marjolein, who is and was always there for me, even if I didn't want anybody to be there, and my parents and parents-in-law who continuously showed their interest in everything I did.

Soli Deo Gloria.

Romke F. Veenstra

April 2017

TABLE OF CONTENTS

1 PROBLEM BACKGROUND AND APPROACH

This master thesis is written for the completion of my Master Industrial Engineering and Management at the University of Twente. The goal of this thesis is to construct the algorithms to be used in a planning system and marketplace for freight orders, called 'Upload!'. On this marketplace transportation orders that are inefficient for the owner of the order can be exchanged with other participants of 'Upload!'. In this first chapter, we start with some general information about the executing company, InfoStructure (Section 1.1). Then in Section 1.2 we describe the proposed service 'Upload!' and the motivation for the project. In the second part of this chapter (Sections 1.3 through 1.7) describe the research design. Finally, in section 1.8 an outline for the rest of the thesis is provided.

1.1 INFOSTRUCTURE

The credo of InfoStructure is "to get from A to B more cleverly using mobile information services". InfoStructure was founded in 2013 in Rijswijk and focuses on a combination of travel and information technology. It offers information services to car drivers, companies and the Dutch government, aiming for more efficient transportation of people and freight. Aim is to increase mobility on the Dutch road network. To achieve this aim, InfoStructure works closely together with 'ARS Traffic & Transport Technology' in The Hague. ARS develops technological solutions for transportation, intended for business and government.

1.2 UPLOAD!

InfoStructure is creating a new service for the logistics sector called 'Upload!', which supports companies in dynamic trip planning and in reduction of the number of inefficient trips by proposing exchange of freight between transportation companies. From the servers of the transportation companies, relevant data is loaded into an optimisation platform. This optimisation platform creates trips and determines which orders are cost wise inefficient. Transportation companies are than given the opportunity to exchange these orders with other companies participating in 'Upload!'.

1.2.1 Motivation

For many transportation companies it is either too expensive or too complicated to purchase a Transportation Management System (TMS). As a result, many smaller companies are planning their trips and loads manually, which might lead to mediocre results. Furthermore, due to the lack of a TMS, these companies might be tempted to except as many orders as possible, without checking whether these orders lead to financial gains. Therefore, these companies might have to perform many inefficient trips without financial benefit.

This group of smaller transportation companies may not be neglected. As we see in Figure 1.1, from the more than 10,000 Dutch transportation companies, over 7,500 of them have only five employees or

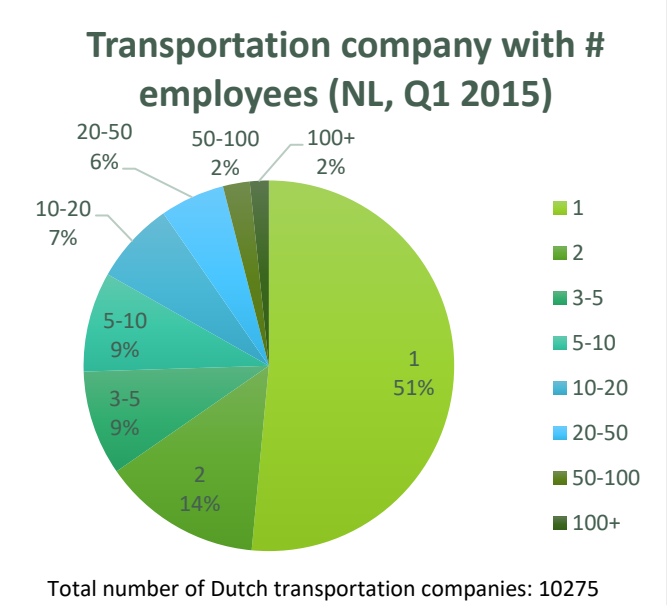


Figure 1.1 Number of employees per Dutch transportation company, 1st quarter of 2015 (CBS1, 2015)

less. These companies often have few options to exchange freight using the existing marketplaces because there is not enough capacity to monitor offered freight. Furthermore, the possibilities to compensate for an inefficient order are smaller because these companies do not have the economies of scale larger companies do have. Overall, by giving this group the possibility to exchange orders with other transportation companies with the same characteristics, 'Upload!' provides a large optimisation potential.

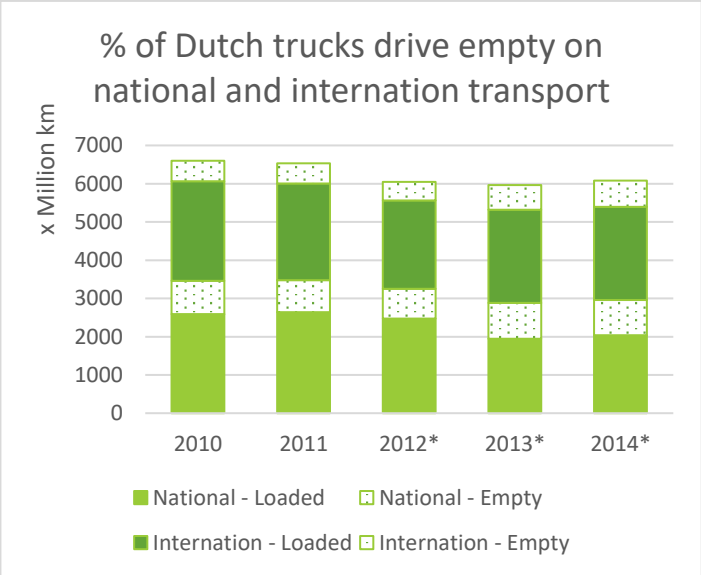


Figure 1.2 Driven distances of Dutch truck both empty and loaded on national and international transport (CBS2, 2015)

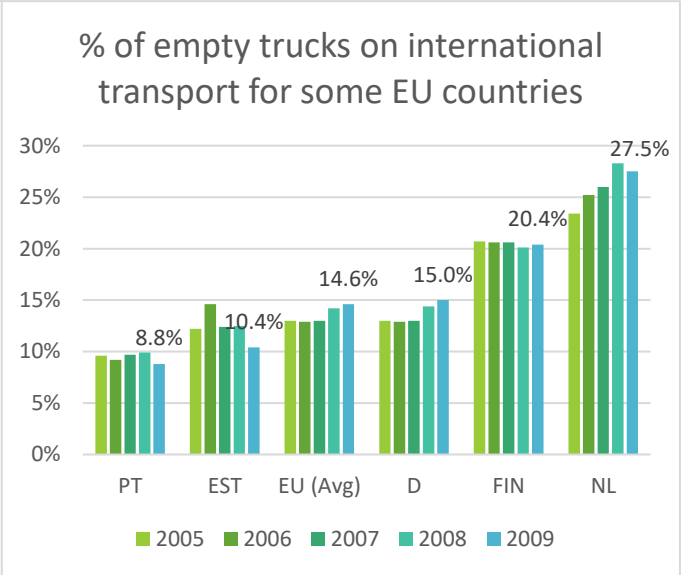


Figure 1.3 Comparison of percentage of empty trips on international transport for some European Union countries (Logistiek.nl, 2010)

Figure 1.2 gives an overview of the distance covered by Dutch trucks, either driving loaded or empty, for both national and international transport. Since 2011, the total distance is declining, possibly due to the economic crisis; whereas the percentage of distance driven empty is increasing. In comparison with other European Union countries, the Netherlands scores worst on the percentage of empty trips on international transport, as shown in Figure 1.3. Both statistics clearly show the potential of an optimisation tool such as 'Upload!'. From the 6 billion kilometres the Dutch companies cover, over 1.5 billion of them are with empty trucks. A reduction of 5% to 6% will already lead to a reduction of around 100 million vehicle kilometres.

Besides empty trips, the load factor (calculated as the used capacity divided by the maximum capacity, either based on weight or on volume) of 'loaded' trucks is an important measure for efficiency. A trip with a load factor of only 20% might be seen as loaded trip (and is seen as such in Figure 1.2 and 1.3), but still might not be efficient if, by means of exchange or outsourcing freight, a higher load factor can be obtained. This is only relevant for less than full truckload (LTL) transport, since container transport has a load factor of either 0 (no container loaded) or 1 (a container loaded), independent of the used capacity within the container. The unused capacity is then a "cost" for the container owner and not for the transportation company.

Every driven vehicle kilometre contributes to emission (amongst others CO_2 and NO_x), and therefore by reducing empty trips and increasing load factors, a reduction in emission is obtained. Overall, 'Upload!' focuses both on empty kilometres and on LTL transport with low load factors, since these trucks can transport freight from other companies. As a result, 'Upload!' cuts both ways: less empty kilometres driven and a higher load factor for non-empty trucks.

1.2.2 Current situation

The proposed solution method by InfoStructure and the target audience is shown in Chapter 2. For now, it is important to know that 'Upload!' must consists of three algorithms, one algorithm to solve dynamic trip planning, one algorithm to determine which orders are inefficient for the company and one algorithm to created exchange proposals of orders between companies. In the remainder of this chapter we discuss the research design.

1.3 RESEARCH MOTIVATION AND GOAL

The motivation for the 'Upload!' project itself is mentioned in Section 1.2.1. This section is about the motivation for the research done in this master thesis.

The goal of this research is to create the algorithms required for the 'Upload!' trip planning and order marketplace. As described in section 1.2.2, 'Upload!' requires three different algorithms for three phases of the problem. First the Vehicle Routing Problem (VRP) for the individual transportation companies must be solved. Various solution approaches for the VRP have been discussed in scientific researches and multiple companies have designed commercial software based on these various solution methods. Selecting an appropriate method is non-trivial and depends on the characteristics of the participating companies.

The solution of the VRP is a set of trips, where a trip is defined as one tour of one vehicle. Even when this solution is the optimal solution for the given set of orders, there still might be so called inefficient orders, orders which largely increase the execution costs of the solution. These orders are for example orders that require are large detour to fulfil and orders that require an additional truck to operate, which otherwise did not have to operate. To determine which orders are inefficient to the owning company, and therefore eligible to offer on a marketplace, a second algorithm with selection criteria is constructed.

Once a list of orders eligible for exchange is created, a third algorithm must determine which order can be exchanged with which other company. In this step the solutions to the VRP are modified to achieve a more global optimisation instead of a local optimisation per transportation company. Some key characteristics of this exchange are calculated to help the accepting company to decide whether to accept the order (examples include the length of the detour and the increased length of the trip).

Overall, we formulate to the following research goal:

***Research goal**: To propose, design and verify the algorithms for the 'Upload!' route planning and matching software.*

1.4 RESEARCH APPROACH

As approach to this research we choose to formulate research questions to reach the research goal. In this section we first formulate these questions, one main question and multiple sub questions. For each of these questions we then determine which methodology and data source we use to answer the question.

We choose the following research question:

***Research question**: How to choose, select, modify and implement models that form the algorithms of 'Upload!'?*

To structure the process of answering this question, we first split the problem into three parts, representing the three steps of 'Upload!': the VRP problem, the selection of inefficient orders and the order exchange algorithm. We formulate one or more sub questions for each part, while all sub questions together fully answer the research question. We show these research questions in Figure 1.4.

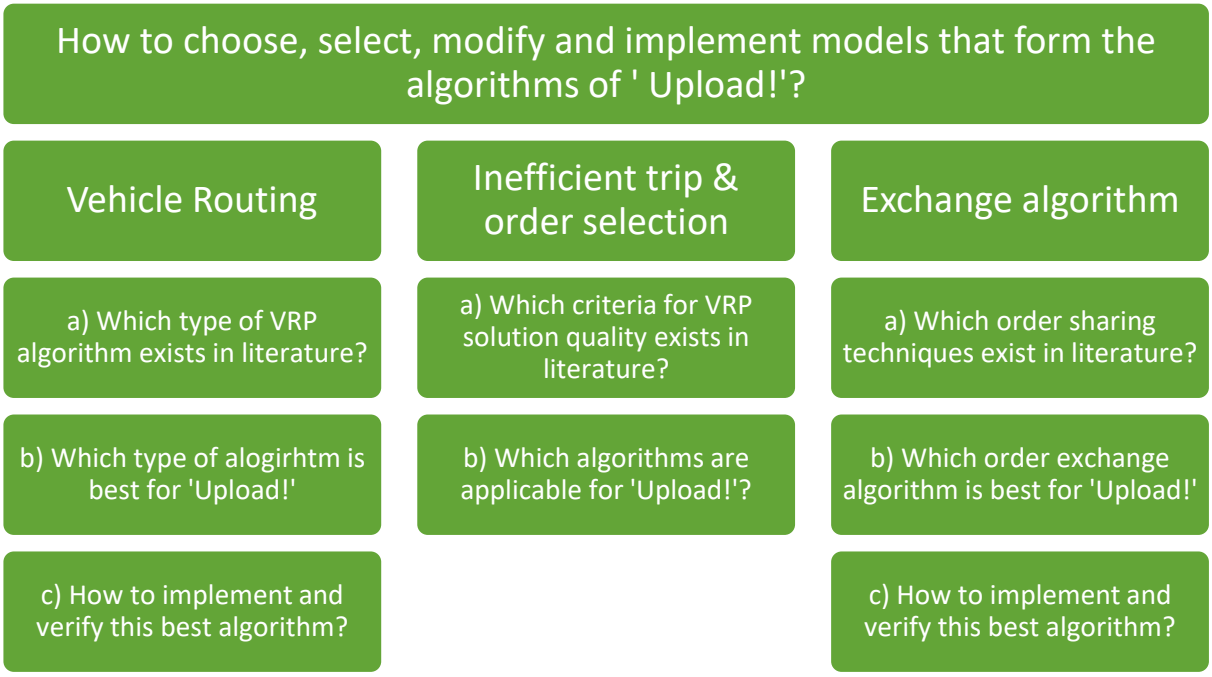


Figure 1.4 Overview of sub research questions

To answer these research questions, we use a combination of different methodologies and data sources. An overview of the used methodology and data source per sub question is provided in Table 1.1.

The first set of research question (1a-1c in Figure 1.4) is about the VRP problem. This consists mainly of literature review on the topic of VRP. Furthermore, the characteristics of the companies targeted to use 'Upload!' will be summarised to determine the best type of VRP algorithm. This step will be executed in consultation with InfoStructure. The implementation of a simplified version of the best performing algorithm is the first modelling part of the research. The validation of this method will be done based on a set of benchmark data for the VRP problem.

The inefficient order selection algorithm forms the link between the VRP and the exchange. To determine which orders cause high transportation costs, a list of decision criteria is constructed based on scientific literature. Based on the expert opinion of InfoStructure and the profile sketch for the participants of 'Upload!' the relevant factors are determined and a selection algorithm is constructed.

The third and final part of the research focuses on the algorithms for the order exchange problem. We start with a literature review on existing exchange and collaboration methods. Based on the existing models and requirements for 'Upload!', we select the most appropriate model, based on a comparison study. Finally, we modify the most appropriate existing model to match with 'Upload!' and implement this using the orders selected by the inefficient order selection algorithm. To validate the selected method, we compare the results of the VRP benchmark before and after exchange of orders.

Table 1.1 Research methodology and data source per sub question

	Subject	Methodology	Data Source
VRP	a) Existing VRP algorithms b) Selection of best c) Model	Literature review Experts opinion .NET implementation	Scientific literature InfoStructure Existing & Benchmark data
Inefficient trip/order selection	a) Existing criteria b) Applicable criteria	Literature review Experts opinion	Scientific literature InfoStructure
Exchange	a) Existing order sharing algorithms b) Selection of best c) Model	Literature review Comparison study .NET implementation	Scientific literature Results of a) Results of order selection

1.5 Research scope

We limit this research to the selection of best algorithms for the individual problems and, to validate the proposed algorithms, a proof of concept based on benchmark data. As such, the actual implementation of 'Upload!' at companies and into apps is not part of this research.

1.6 Deliverables

In this section, the deliverables of the research are given. Some of the deliverables are part of the thesis, whereas the pilot implementations of the proposed models are delivered to InfoStructure separately.

- The master thesis
  - Selection and description of validated VRP solving algorithm with parameter determination.
  - Description of inefficient order selection algorithm.
  - Selection and description of algorithm for the order exchange problem.
  - Conclusions of this research and recommendations for further improvements and research.
- Implemented VRP solving algorithm as proof of concept and data source for order selection algorithm and order exchange algorithm.
- Validated model to solve the order exchange algorithm, implemented as a proof of concept

1.7 Thesis Outline

In the remainder of this thesis we describe the context of 'Upload!' in Chapter 2 and the relevant scientific literature regarding the VRP and order exchange in Chapter 3. In Chapter 4 and 5 we discuss the solution methods and the numerical experiments done with these methods and finally in Chapter 6 the conclusions and recommendation for further research.

2 CONTEXT ANALYSIS

In this chapter an overview of the context of 'Upload!' is given. In Section 2.1 we describe the target participants of 'Upload!' and their specific characteristics, both from the vehicle routing point of view as well as from the exchange potential point of view. In Section 2.2 we describe how the planned solution looks like and which global characteristics it must have. In Section 2.3 these characteristics are translated into functional specifications and finally in Section 2.4 a roadmap to implementation is shown.

2.1 TARGET PARTICIPANTS FOR 'UPLOAD!'

Since we are still in the design phase of 'Upload!', there are not yet participants. As such it is difficult to give a complete description of 'Upload!' participants, although, from the target participants a sketch can be made. Typical customers of 'Upload!' might include smaller transportation companies for parcel delivery in cities, replenishment of DIY shops and offices and replenishment of pubs and bars. In the long run the VRP part might also be used for WMO (Dutch social support act) transportation.

To determine which participant characteristics are important for the design of 'Upload!', we describe the target participants based on three main characteristics, namely the vehicle fleet, the orders and the management structure.

Vehicle fleet

'Upload!' is designed for smaller transportation companies, typically with few (up to 5) trucks. Many of these companies do not have an existing TMS or the time to communicate with other transportation companies to exchange freight. This small amount of trucks makes it important for the VRP algorithm to reduce the number of trucks used as much as possible. On the exchange part this limits the total amount of orders involved in the marketplace, since there are restrictions on number of vehicles.

Furthermore, the vehicle fleet for these smaller companies will mostly be homogeneous with respect to the freight type. This is important for the selection method for the VRP. The vehicle types may however vary between the various participants, e.g. vehicles for parcel delivery in cities are probably different compared to vehicles for barrels to pubs and bars. In the exchange phase this is important since it limits the possibilities to exchange freight, because a suitable truck from another company must be found.

Order characteristics

All companies that are going to use 'Upload!' are transportation companies; they transport freight or people from a pickup location to destination. This is different from the standard VRP where freight is transported either from a depot or to a depot. The pickup and delivery VRP is well documented in scientific literature however, so the literature research can be focussed in this direction.

Another order characteristic is the use of time windows. Since the freight is transported by a transportation company, arrangements between the transportation company and the freight owner are made about time windows, a start and end time between which the freight must be picked up or delivered. In case of city delivery these time windows are in general rather small, since there are regulations about store replenishment times in cities and there is a commercial interest in the right delivery time.

Finally, for city deliveries the distances between customers are often small (clustered orders) and driving times depend largely on the time of day (rush-hours or not). Especially for parcel delivery, the freight size is usually small, so many orders fit in one truck.

Management structure

Smaller companies usually have not much experience with order exchange and the expectation of InfoStructure is that they might be afraid of losing orders to concurring companies. As such, regulations about which company can exchange orders with which other company must be part of the exchange algorithm. Furthermore, there might also be the need of a one-for-one exchange mechanism where orders are only exchanged if both companies remain having the same number of orders.

2.2 PLANNED SOLUTION OVERVIEW

As explained in Chapter 1, 'Upload!' will consists of three parts; the VRP algorithm, the inefficient order selection algorithm and an exchange algorithm. Figure 2.1 shows how each of these algorithms functions and how they cooperate.

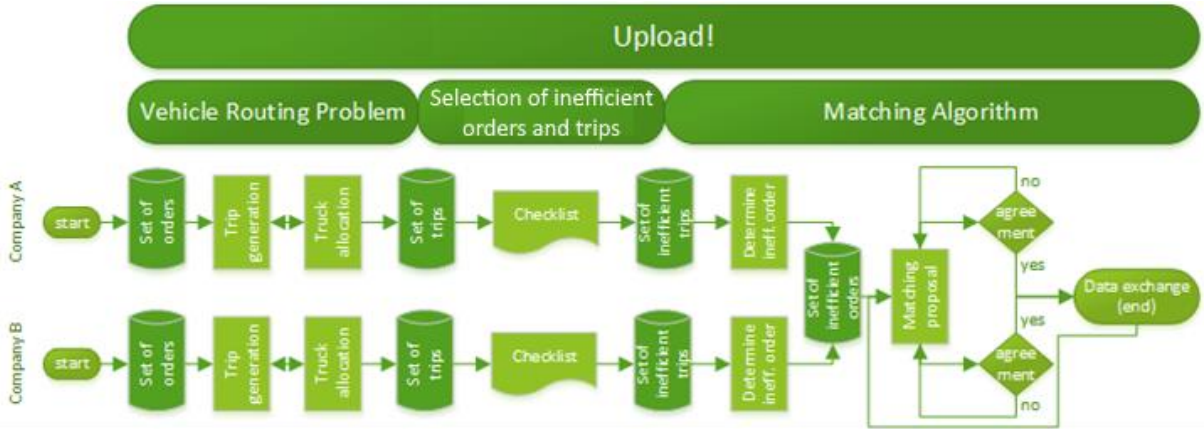


Figure 2.1 Overview of Upload!

Each of the steps is further explained in Table 2.1 below. The functional specifications as prescribed by InfoStructure are mentioned in Section 1.2.3.

Table 2.1 Description of steps of the 'Upload!' overview

Set of orders	Every company delivers a set of orders that must be planned.
Trip generation & Truck allocation	The trip generation, together with the truck allocation, this is the first required algorithm. Optimal trips based on the input orders are generated and allocated to the right available truck. This is an iterative process.
Set of trips	The result of the VRP algorithm is a set of trips
Checklist	The characteristics of every trip are compared to a threshold value. Inefficient trips are selected.
Set of inefficient orders	The set of inefficient trips is used as input for the second step of the selection algorithm, the determination of inefficient orders.
Determine inefficient orders	In this algorithm the orders causing inefficiency are selected based on the cost doing the associated pickup and delivery.
Set of inefficient orders	The resulting set of inefficient orders is the input for the matching algorithm. This is a collection of all inefficient orders of all participants of 'Upload!' .

Matching proposal	From the set of inefficient orders matches between companies about exchanging orders are proposed. Many matching proposals might be produced in this step.
Agreement	Either using manual supervision or automated based on the characteristics of the exchange, the companies involved in the match can accept the proposal.
Data exchange	If companies agree to exchange the orders, the necessary information is exchanged between the participants.

2.3 DATA SOURCE

As the main source of data, a benchmark will be used to test the various results of this research. Li and Lim (2001) introduced a set of instances for the Pickup and Delivery VRP with time windows (PDVRPTW) that seems to have become a standard benchmark set for this problem (Ropke, 2005). The instances where constructed based on the Solomon's test for the VRP (Solomon, 1987) for the instances with 100 order points which is used in this research. The instances where created by first solving the Solomon's problems for the VRPTW (so without the pickup and delivery part) and then pairing order points which happened to be on the same route in the VRPTW solution. The requests are paired such that the pickup is visited before the delivery. A disadvantage of this method is that the instances might not result into very realistic instances. A complaint might be that the instances constructed in this way are too easy because pickup and delivery locations fit too well together since they are constructed based on the solution of the Solomon's benchmark (Ropke, 2005). According to the same research of Ropke (2005) however, it turns out the instances created this way "challenging for both heuristics and exact methods, especially for larger instances with 100 requests or more".

The Li Lin benchmark consists of six different categories, varying in order point layout and vehicle capacity. Clustered order points mean that a number of pickup and delivery locations are clustered around a central point. This is typical for e.g. city delivery in real live applications. Random order point layout means that the pickup and delivery point are not necessary in the same area and that there are no clusters of order points. Finally, the mixed order point layout mixes the clustered and random layout. These three types of order point layouts are also used in the Solomon's benchmark. Besides order point layout the instances also vary in vehicle capacity, which is used to vary the number of trucks required to execute the whole set of orders. Higher vehicle capacity generally corresponds with a best-known solution of 3 to 4 trucks, whereas a low vehicle capacity corresponds with a best-known solution involving up to 20 trucks.

Table 2.2 Type and number of instances in Li & Lim's PDVRPTW Benchmark

Type	# of instances	Order point layout	Vehicle capacity
LC1*	9	Clustered	200
LC2*	8	Clustered	700
LR1*	12	Random	200
LR2*	11	Random	1000
LRC1*	8	Mixed	200
LRC2*	8	Mixed	1000

When combining the characteristics of the target participants of 'Upload!' with the benchmark we see that the instances LC2 and LRC2 best fit these characteristics. As such in the remainder of this thesis we will use the LC2 and LRC2 benchmark to test our results.

2.4 FUNCTIONAL SPECIFICATIONS

InfoStructure lists a set of functional specification for the 'Upload!' platform. These specifications can be divided into two groups. On the one hand, there are the specifications for the VRP and on the other hand there are the specifications for the exchange algorithm. We first enumerate the specifications for the VRP:

- Dynamic trip planning
  - Additional orders during the day
  - Stochastic driving times (driving time can depend on time of day, e.g. rush hour).
- Trip creation with up to 23 customers per trip.
- Restrictions from transporting company and legislation regarding drive and rest-times.
- Time windows (start and end time between which the order must be delivered).
- Connection with fleet capacities to ensure capacity optimization.
- Real time visualisation of all vehicles with load factors.

For the exchange algorithm, we enumerate the following specifications:

- Matching option either to:
  - Fill empty or partially loaded trips with freight from other transportation companies.
  - Reduce number of empty or partially loaded trips by offer freight to other transportation companies.
- Option to accept a match which leads to exchange of orders between companies.
- Creation of new trip planning including exchanged order.
- Overview:
  - Of additional freight per transportation company due to order exchange.
  - Of offered freight to other transportation company due to order exchange.
  - Of impact of new trip planning on:
    - Load factor.
    - Vehicle kilometres.
    - Travel time (including travel time during rush hours).
    - CO_2 emission.

2.5 ROADMAP TO IMPLEMENTATION

InfoStructure proposes the following roadmap to the implementation of 'Upload!':

1. Design communication and data management platform (Enterprise Service Bus or ESB)
2. Connect and design libraries with specific constraints
3. Adjust ARS mobility platform to become suitable for ESB
4. **Design of algorithms (VRP, inefficient order selection and exchange)**
5. Design mobile apps for Android and iOS
6. Connection mechanism between databases of transportation companies
7. Attract transportation companies and start a test phase
8. Adjust 'Upload!' with suggestion of transportation companies
9. Roll out 'Upload!' as system for at least 500 transportation companies and shippers.

For this graduation assignment only part 4 of the roadmap is relevant.

3 LITERATURE REVIEW

This chapter describes the current state of literature to our research. In section 3.1 the used methodology is summarized, after which in the sections 3.2 and 3.3 the relevant literature regarding the Vehicle Routing Problem and the Matching problem is presented. The latter two sections are constructed based upon the methodology of the first section.

3.1 REVIEW METHODOLOGY

To properly perform this literature review, the methodology as proposed by Heerkens (2005) is used. Aim of the literature study is to solve the so-called knowledge problem for the research. The solving method of Heerkens consists of a few steps, which is followed to answer both the knowledge problems for the VRP as well as the knowledge questions for the Matching Algorithm. These steps are summarised below:

1. Identifying the knowledge problem and formulate problem statement;
2. Dividing the knowledge problem into knowledge questions;
3. Determining the steps required to solve the knowledge problem;
4. Perform the steps required;
5. Process the found literature and answer the knowledge questions;
6. Answering the knowledge problem.

3.2 VEHICLE ROUTING PROBLEM

In this section the research questions as formulated in Section 2.2 regarding the Vehicle Routing Problem are answered. The research questions “Which type of VRP algorithms exists in literature” and “Which type of VRP algorithm performs best for ‘Upload!’” contain a knowledge question. These two research questions are (step two of the method of Heerkens) translated into five knowledge questions, which are answered using a literature study into the field of VRP. These questions are shown below and are answered in section 3.2.1 through 3.2.5:

1. What is a Vehicle Routing problem?
2. Which type of Vehicle Routing problems exists?
3. Which solution algorithms for the VRP exist?
4. Which solution algorithm works best in which situation?
5. Which commercially available software exists and how do they compare?

The fourth and fifth literature research question, combined with the requirements for ‘Upload!’ is the input to determine which commercial VRP algorithm is recommended for InfoStructure to use. A derivative of this model is implemented in this thesis.

3.2.1 What is a Vehicle Routing Problem?

The VRP was first introduced by Dantzig and Ramser (1959) as a generalisation of the Travelling Salesman Problem (TSP) in order to obtain a routing schedule for petrol deliveries. Aim of a VRP is to construct an optimal delivery or collection route between multiple locations starting and ending from one or more depots. A mathematical formulation of the classical VRP with one depot is as follows (Laporte, 1992):

The classical VRP is defined on a graph $G = (V, A)$. Here $V = \{1 \dots n + 1\}$ is a set of vertices representing n pickup locations; with the depot located at vertex 1, and A a set of arcs between

locations. For every arc (i,j) $i \neq j$, a non-negative cost factor (c_{ij}) is determined and associated in a cost matrix C. The value c_{ij} can be either travel time, travel distance or a cost factor for travelling this distance. When C is symmetrical ($c_{ij} = c_{ji}$), it is convenient to replace the set of arcs A by a set of undirected edges (E). The number of vehicles m operating from a depot can either be fixed or free (free means m can be chosen between an upper bound m_U and lower bound m_L such that $m_U \neq m_L > 0$). When m is not fixed, it often makes sense to associate a fixed cost f on the use of a vehicle. The VRP consist of designing a set of least-cost vehicle routes in such a way that:

i. Each city in $V \setminus \{1\}$ is visited exactly once by exactly one vehicle;
ii. All vehicles start and end at the depot;
iii. Additional side constraints.

Without any additional side constraints, the formulation for the VRP is equal to that of the TSP, since every location can be visited by the same vehicle, i.e., the shortest route between multiple locations must be found, as is the TSP. The side constraints determine the type of VRP, see Section 3.2.2. Various solution methods for different kinds of VRP's are discussed in Section 3.2.3.

3.2.2 Which type of Vehicle Routing Problems exists?

The VRP as introduced by Dantzig and Ramser (1959) had as side constraint the capacity of the petrol tanker and as objective to minimise the total route length. Over time, multiple variants of this initial problem have been studied. A broad field of constraints, features and conditions from practice are applied to the VRP (Weise, Podlich, & Gorldt, 2009). A framework of the various types of VRP's is shown in Figure 3.1. Besides these types of VRP's, the objectives of the optimisation may also depend from situation to situation. The different objectives are shown in the last part of this section, under the heading "Objectives".

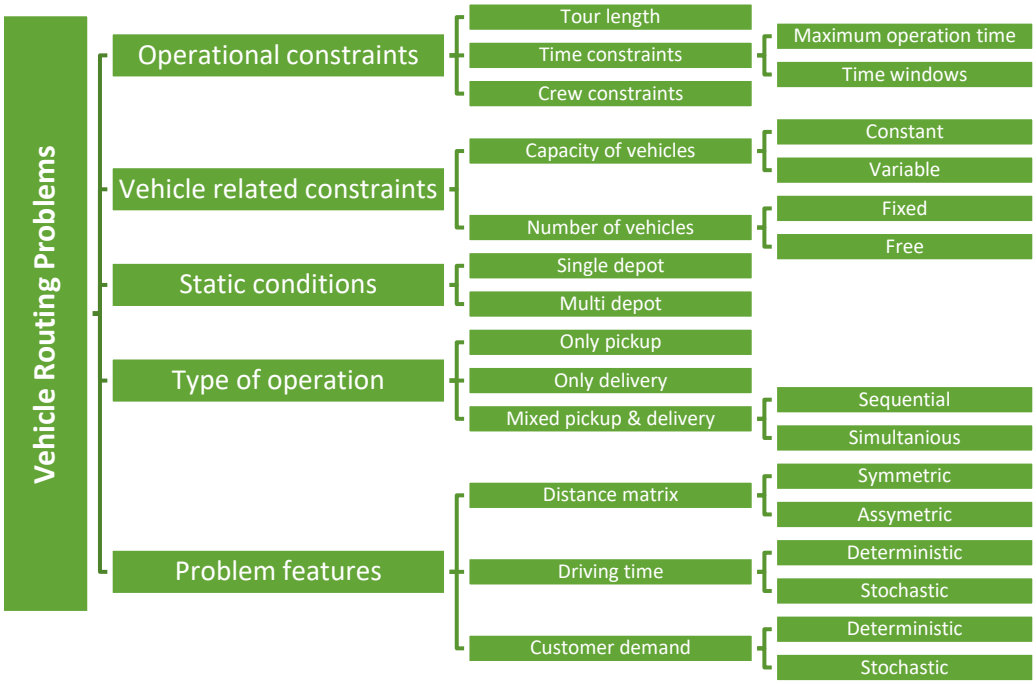


Figure 3.1 Framework of constraints, features and conditions applicable for the VRP. Based on Ganesh, Sam Nallathambi, and Narendran (2007) in (Anbuudayasankar, Ganesh, & Mohapatra, 2014)

Operational constraints

For real life applications of the VRP many operational constraints might occur. These constraints may either be enforced by government or local authority, set by the customers or be based on the

preferences and capabilities of the crew. According to Anbuudayasankar et al. (2014) the most important operational constraints are:

- Transportation companies set a **maximum tour length**, which is either a maximum distance or a maximum number of locations per trip, in order to maintain flexibility. (Laporte, 1992)
- Government regulations regarding **maximum** adjacent, daily and weekly **working hours** for truck drivers. Regulations depend on the local government; the Dutch regulations follow European regulations. (VerkeerWaterstaat, 2010)
- Customer constraints include delivery **time windows**, one or more time frames per day during which delivery or pickup can be performed (Laporte, 1992). Poot, Kant, and Wagelmans (2002) describe the situation where there are two timeframes per day, e.g., at the start and end of the opening time of a shop. Time windows might as well be forces by governmental loading regulations at the customers' location.
- Truck drivers might be only familiar with a specific locations or regions and as such it is preferred that drivers are assigned to trucks serving these locations or regions. This constraint is also called the **region constraint**. (Poot et al., 2002)
- Various other constraints including specific licence for truck drivers and limitations in the availability of goods at the depot. (Poot et al., 2002)

Vehicle related constraints

Typical observed constraints regarding the vehicle are:

- The truck capacity. When vehicles are capacitated, which is in most practical situations intuitively true, the capacity may not be exceeded. The vehicle fleet can either be homogeneous; every truck has the same capacity, or heterogeneous; different trucks with different capacities. In the latter situation an additional decision, which trip to assign to which truck, must be made. (Laporte, 1992)
- Number of vehicles. The number of vehicles is either fixed (predetermined) or free (the number of vehicles might vary from day to day). In this situation a fixed cost of using a vehicle is applied in order to minimise the number of vehicles used. (Toth & Vigo, 2001)

Furthermore, many additional vehicle-related constraints might apply. This is highly company specific and makes it difficult to incorporate these in a generic model. These constraints include: (Fisher, 1995)

- Specific goods must be transported with specific trucks.
- Specific goods may not be transported together with other freight.
- Every visiting location is assigned to one or more vehicle compartments.
- A vehicle might be capable of performing more than one trip within the planning horizon.

Static conditions

Decision made by the transportation company on a strategic level, such as the number of depots, for the static conditions of the VRP. Since decisions made on a strategic level are long term decisions, these conditions form the outline of the routing problem (Montoya-Torres, López Franco, Nieto Isaza, Felizzola Jiménez, & Herazo-Padilla, 2015). Problems with one depot, i.e., single depot problems, are well known in literature. Multi-depot problems on the other hand are studied less frequent. The multi-depot aspect adds an additional layer of complexity and decisions, and according to Ho, Ho, Ji, and Lau (2008) this makes the multi-depot VRP difficult to solve to optimality, even for small problem instances.

Type of operation

The basic operation executed at the locations depends from company to company. Most problems in literature deal with either **pickup or delivery**. When pickup and delivery are handled in the same trip, either one operation can precede the other (all pickup locations must be visited before any delivery location can be visited or vice versa, also called **sequential operation**) or the operations are mixed, also called **simultaneous operation**. In case of mixed pickup and delivery, it is possible to further distinguish between few-to-many (few pickup locations with high loads and many delivery locations with smaller loads), many-to-many, many-to-few/one (e.g., pickup locations of patients to one central hospital) or one-to-one (e.g., taxi rides).

Problem features

The VRP can either be deterministic, i.e., all input data (order specifications, vehicle specifications and driving times between locations) is known when the trips are designed, or stochastic, if one or more of the input data is not known in advance. Often the classification static or dynamic is used, see Figure 3.2. Static VRPs are VRPs where the order specifications are known upfront. In the case of varying orders, i.e., dynamic VRPs, either additional orders might be added during operation or the customer demand or supply is not known in advance, in which case the capacity of the vehicle might be reached at an unknown moment.

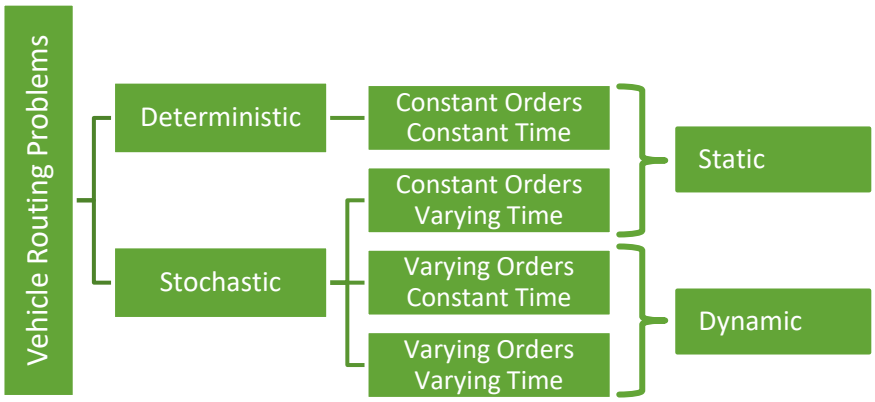


Figure 3.2 Classification of problem features (Ganesh et al., 2007) in (Anbuudayasankar et al., 2014).

Objectives

As shown in the mathematical description of the problem, the cost matrix C might indicate different costs. Since the objective is to minimize the total cost function, this gives room for different variants of the problem. Some common used objective functions for the VRP are: (Toth & Vigo, 2001)

- Minimise the **total** route **length** (C indicates distance between locations).
- Minimise the **total** travel **time** (C indicates travel time between locations).
- Minimise the **total** transportation **costs** (C indicates cost to travel between locations).

Besides the objective in the VRP formulation, other objectives may exist including, but not limited to:

- Minimise the **number of vehicles** needed to serve all locations. (Toth & Vigo, 2001)
- Minimise the **variation** in **travel time** and **vehicle load**. (Toth & Vigo, 2001)
- Maximise **customer satisfaction** (minimise incurred penalties). (Toth & Vigo, 2001)
- Maximise **visual attractiveness** of constructed trips (the more attractive a solution looks to the planner, the more likely that the planner will accept the generated solution). (Poot et al., 2002)

For most VRP applications one of the common objective functions is used and none, one or more of the less common objectives are used.

3.2.3 Which type of solution methods exists for the VRP?

Due to the large variety of different types of VRPs and the broad field in which the problem is stumbled upon, many different solution methods have been proposed over the course of the last fifty-five years since the introduction by Dantzig and Ramser in 1959 (Montoya-Torres et al., 2015). As shown in Figure 3.3, solution approaches to the VRP can be classified into five broad categories. Within each category different algorithms have been modified to suit the needs for a particular type of problem.

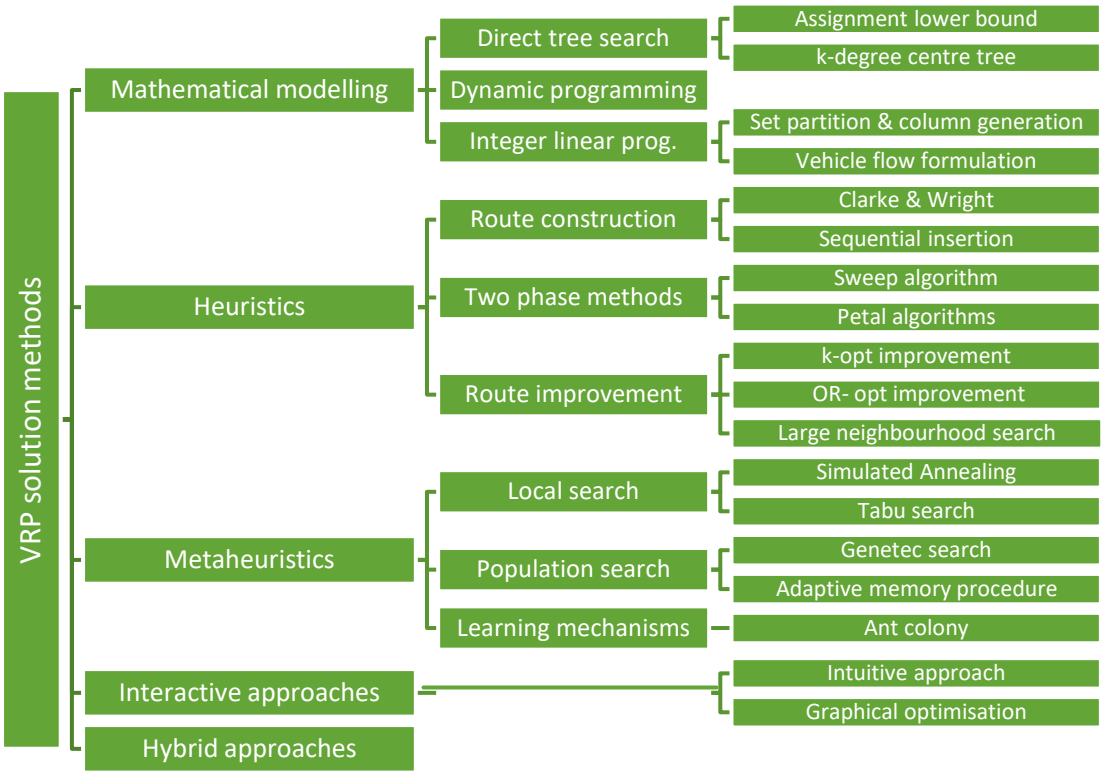


Figure 3.3 Overview and categorisation of VRP solution methods (Brotcorne, Laporte, & Semet, 2002; Cordeau, Gendreau, Laporte, Potvin, & Semet, 2002; Fisher, 1995; Laporte, 1992)

Mathematical modelling (Exact algorithms)

Mathematical modelling ensures that the exact optimal solution to the problem is found. This is a large advantage over alternative methods, which do not ensure optimality. The classical VRP is known to be NP-hard, since it generalizes both the TSP and the Bin Packing Problem (BBP), both of which are well known NP-hard problems (Garey & Johnson, 1979). As such, the computation time for exact solutions increases strongly with the size of the problem, the number of locations to be visited. Therefore, especially for real world instances of the VRP, heuristics are used.

Exact algorithms can be divided into three classes as shown below. For each class, we show a couple of typical examples. These methods were often initially used to solve the TSP and later modified to meet the constraints of the VRP (Laporte, 1992).

1. **Direct tree search** methods – The problem is branched and using increasingly better lower bounds the solution space is reduced. Examples include:

- **Assignment lower bound** and a related branch-and-bound algorithm – uses a relaxation to reduce the problem to a TSP with multiple vehicles (m-TSP), which can be further reduced to a classic TSP (1-TSP) by introducing m copies of the depot which must be visited, where m equals the number of vehicles in the VRP.
- **The k-degree centre tree** and a related algorithm – Also relaxes the problem to an m-TSP but uses the depot as the centre of a k-degree tree, where k equals the number of vehicles in the VRP. Both methods are most suitable for a fixed number of vehicles and therefore a fixed number of trips.

2. **Dynamic programming** (DP) – Eilon, Watson-Gandy, and Christofides (1971) proposed a DP method to solve the VPR. The number of states however must be reduced using a relaxing method because otherwise the number of computations will be excessive for most problems.
3. **Integer linear programming** (ILP) – A variant of the well-known linear programming technique where as an additional restricting all variables must be integer numbers. Some methods using ILP include:
    - **Set partitioning and column generation** – Arguably the best known exact solution method for the VRP, a full set of all possible partial solutions (ordered lists of locations) is generated and a decision on whether to include an item of this set is a binary decision. This method works best for strongly restricted problems (e.g., VRPs with time windows) since the set of potential solutions is limited.
    - **Vehicle flow formulation** (VFF) – In the three index VFF binary variables x_{ijk} indicate whether the arc (i, j) is driven by vehicle k. In addition, a binary variable y_{ik} indicates whether location i is served by vehicle k. Using this notation an ILP can be constructed.

Heuristic algorithms

Since exact algorithms often have high computation time, they are often not suitable for real world applications. Heuristics are used to find a feasible, not necessarily optimal, solution for the VRP. As for the exact algorithm, heuristics are often based on existing heuristics for the TSP, modified for the constraints of the VRP (Laporte, 1992). Heuristics for the VRP can be divided into three categories. For each of these heuristics some typical examples are shown below (Brotcorne et al., 2002):

1. **Route construction** heuristics – as the name suggests are used to construct a set of feasible routes. The most well-known route constructions are:
    - **Clarke & Wright savings algorithm** the first heuristic suggested for the VRP and still, even though introduced in 1969, used by many companies. It is based on the idea of merging two existing routes into one new, combined route. The reduction in distance due to the merge is called a saving. During every iteration of the heuristic the highest possible saving is executed and the algorithm continues until no further savings can be made.
    - The **sequential insertion algorithm** forms a trip starting with the most difficult location to add (in practice often the farthest location not yet on a route) and inserting other locations based on their difficulty. This method is often combined with a route improvement heuristic. (Joubert & Claasen, 2006)
2. **Two-phase heuristics** – these algorithms split the VRP into two parts, first cluster locations into subsets corresponding with routes and afterwards determine the sequence in which the locations are visited. Some examples include:
    - **Sweep algorithm** is the first example of a two-phase algorithm. The clustering is based on one seed location and adds additional locations in increasing polar angle

from the depot until capacity is reached. The solution to this method highly depends on the choice of the seed location.

- **Petal algorithms** generate a large set of feasible routes, known as petals, and select a subset of these routes using a set partitioning model. This model is related to the exact set partitioning and column generation model, although not all feasible routes are generated. These algorithms perform in general superior compared to the sweep algorithm.

3. **Route improvement heuristics** – these heuristics improve the quality of an earlier created route and are often used as final step of either another heuristic or a meta-heuristic.
    - **k-opt improvement algorithms** improve a generated set of routes by exchanging λ locations between routes. Since the number of possible exchanges equals n^{λ} usually only $\lambda = 2$ or $\lambda = 3$ is used.
    - **OR-opt improvement algorithms** also use the principle of generation by exchanging, as does k-opt, although instead of exchanging 2 or 3 locations, a chain of consecutive costumers within the same trip is relocated.
    - **Large Neighbourhood Search (LNS)** is an algorithm to move from one potential solution to another potential solution by demolishing and regeneration of larger parts of the solution. As such it is possible to escape local minima and move from one set of potential solutions to another set.

Metaheuristics

According to Gendreau, Potvin, Bräumlaysy, Hasle, and Løkketangen (2008), meta-heuristics are the most promising and effective solution methods for the VRP. Metaheuristics perform a scattered and more thorough search of the solution space and are less likely to end in a local minimum compared to heuristics (Cordeau, Laporte, Savelsbergh, & Vigo, 2006). Metaheuristics often make use of a neighbourhood structure and can temporary accept a less optimal solution to escape local minima. The quality of solutions produced using a metaheuristic are often better than that of heuristics; which comes with the price of increased computation time (Anbuudayasankar et al., 2014).

Metaheuristics can be classified in various ways, but for this research we used the classification of Cordeau et al. (2006), who divides the metaheuristics into three types:

1. **Local search** methods – these methods improve the local search heuristics by probabilistically accepting less optimal solutions to escape local minima.
    - **Simulated annealing** is a local search algorithm where a start solution is drawn randomly from the total solution space. Simulated annealing is based on the technique of annealing used in metallurgy, where controlled cooling is uses to increase the quality of a material. In simulated annealing, the concept of cooling is used to find a good solution. In the beginning the temperature (variable) is high and many solutions are accepted and explored; whereas as the temperature decreases the probability of accepting less optimal solutions is also decreased.
    - **Tabu search** used the concept of a taboo list which contains already visited neighbourhood structures to avoid cycling. From any starting solution Tabu search moves to the next best available solution satisfying all constraints.
2. **Population search** methods – metaheuristics that mimics the process of natural selection to generate new solutions. It uses crossover operations which combines two parts of the parents and then apply a mutation operation to each offspring.

- **Genetic search** is a method that searches the best available parent solutions among a pool of candidates and combines the best parts of them. Crossover and mutation operations for the multi depot VRP are described by Ho et al. (2008).
- **Adaptive memory procedures** vary from genetic search in that sense that the best parts from multiple solutions, not just the two parents as in genetic search, are taken and combined to create new solutions.

3. **Learning mechanisms** – these mechanisms use information gained in previous iterations to better perform (learn) operations in the next iteration.
    - **Ant colony optimization** is another nature inspired algorithm, based on the ability of ants to find the shortest path between food and their nests. Ant colony optimization combines a savings based heuristic with a 2-opt improvement procedure to generate feasible solutions. In every next solution, the savings are modified to take the attractiveness between two locations in previous solutions into account.

Interactive approaches

This is a special kind of solution methods where human interaction (intuition) is required to solve the VRP. The model aids the planner in the decision making. The quality of such a method cannot be calculated, since it depends on the skills and experience of the planner. (Anbuudayasankar et al., 2014)

Hybrid approaches

This approach combines two or more of the above approaches to further optimise or speed up operations. Ho et al. (2008) uses this approach in a hybrid genetic algorithm (a meta-heuristic), where initial solutions are constructed using a saving method and a nearest neighbour heuristic (both heuristic algorithms). Some hybrid approaches have reported high potential to provide good solutions at low computation time (Anbuudayasankar et al., 2014; Ho et al., 2008). During the last years, more and more research is focussed on the hybrid approach. From an intellectual point of view, however, it does not really add new methods, only new combinations of methods.

3.2.4 Which type of solution algorithm works best in which situation?

For the assessment of the quality of a solution method for the VRP we use the four attributes proposed by Cordeau et al. (2002), namely accuracy, speed, simplicity and flexibility. The latter two are often neglected, but are, from a company's point of view, important with respect to the options to implement and maintain the models. In Table 3.1 a ranking of the various solution methods based on these four attributes is provided.

Accuracy is the measure for deviation between the optimal value and the solution value of a heuristic. Accuracy is a trade-off with the **speed** of the algorithm (where speed equals the computation time required for the solution method) since more invested time will almost always lead to better solutions. For many applications however, speed is important, especially in real-time applications. Long-time planning, e.g. to determine fleet size, on the other hand allows longer computation time, especially if large sums of money are involved.

Simplicity is a measure for how easy to understand and implement a method is. This is not only important for the initial construction, but also for the maintenance and parameter settings. If parameters have no intuitive meaning for tactical planners, often non-optimal parameter settings are used. Simplicity tends to strive with **flexibility**, since flexibility is the measure to which degree the method can be modified to incorporate additional constraints and objectives. More flexibility in general leads to less specific algorithms, which are harder to implement therefore less simple.

Table 3.1 Ranking of various VRP algorithms on accuracy, speed, simplicity and flexibility

		Accuracy	Speed	Simplicity	Flexibility	Comments	Source
Exact	Direct tree search	++	--	--	-		2
	Dynamic Programming	++	--	-	-		2
	Integer Linear programming	++	--	-	-		2
Heuristics	Savings algorithm	-	++	++	-	Flexibility deteriorates accuracy	1
	Sequential insertion	+/-	+	+	+/-		3
	Sweep algorithm	-	+	++	--		1
	λ-opt improvement	*	*	++	++	No route construction heuristic	1
Metaheuristics	Simulated Annealing	+	-	+	+		2
	Tabu search	+	+/-	+/-	+	Many different implementations	1
	Large Neighbourhood search	-	+	+	+/-	Combine with other methods	3
	Genetic search	++	-	+/-	+/-		2
	Adaptive memory processing	++	-	+/-	+		1
	Ant colony optimisation	+	-	+/-	+		2
I.A.	Intuitive approach	*	*	++	++	Depends on manual input	
	Graphical optimization	*	*	-	+	Depends on manual input	

1 Cordeau et al. (2002) *2 Anbuudayasankar et al. (2014)* *3 Joubert and Claasen (2006)*

3.2.5 Which commercially available software exists and how do they compare?

Every year OR/MS Today, a part of INFORMS (Institute for Operations Research and the Management Sciences) conducts a survey among the largest vendors of Vehicle Routing software. Vendors supply answers to many questions about their software packet and, thus, a comprehensive overview of the capabilities and characteristics of these software packets is obtained. In Table 3.2 an overview of the most important characteristics of the 15 surveyed software packets is shown.

The results of this survey are difficult to compare against each other since the vendors typically describe a general implementation of their software and not a specific solution which would fit the needs of 'Upload!'. For implementation of commercial software, a specific contract with a supplier, preferable one with much experience in the field of solution generation for smaller companies, must be selected and a custom contract must be created. Here the financial aspect is also of importance, but hard to assess without requesting custom contracts from different suppliers.

Since the types of algorithms used as solution method are mostly proprietary and or a combination of multiple, it is difficult to determine a good solution method based on the algorithms used by commercial companies. Also, the computation time indicates that various methods for various problem sizes are used, since e.g. sub second solutions are only possible in selective cases; not to find a near-optimal solution for a large problem instance.

Table 3.2 Overview of available Vehicle Routing Software packets with specifications

Product	Year	Land	Maximum problem size			Performance		Installed Base
			Number Stops	Number Vehicles	Number Terminal	Computation time*:	types of algorithms	
ArcGIS for TransportationAnalytic	2012	USA	No Limit	No Limit	No Limit	< 5 minutes	6+ core algorithms	
Descartes Routing, Mobile & Telematics	1995	CA	not limited	not limited	not limited	sub second	Holistic & Heuristic	1001+
DISC	1990	UK	unlimited	unlimited	unlimited	a few seconds	MJC2	101-500
eRoute Logistics	2002	USA	no limit	no limit	3	< 4 minutes	Various Heuristics	1-100
JOpt.SDK	2006	D	virtually unlimited	virtually unlimited	virtually unlimited	10 minutes	construction, SA, GA	1001+
Logistics Optimizer	2005	USA	600 stops	200 vehicles	unlimited per SaaS	5 minutes using SaaS	Genetic	1-100
logvrp	2010	TK	web app 500, web API unlimited	web app 100, web API unlimited	unlimited	~30 minutes, depends constraints	LS, heuristics, SA, hybrid	1-100
Optrak vehicle routing software	1992	UK	No fixed limit	No fixed limit	No fixed limit	5-20 mins	Various heuristics	1-100
ORTEC Routing and Dispatch	2003	NL	no limit	no limit	no limit	top 3 performance worldwide	several algorithms, case based	101-500
Paragon Routing and Scheduling Optimizer	1997	USA	20	3	2	Around 2 minutes	Cost saved & improvement algorithms	101-500
Roadnet Transportation Suite	1983	USA	unlimited	unlimited	unlimited	<30 seconds	Proprietary heuristics	1001+
Routist.com	2013	I	unlimited	unlimited	unlimited	3 to 5 mins on our elastic cloud	proprietary meta-heuristic	1-100
StreetSync Desktop	2005	USA	unlimited	unlimited	unlimited	Several minutes	Proprietary	101-500
TMW Appian DirectRoute	1996	USA	1 million	100	Unlimited	< 3 minutes	Proprietary	501-1000
TruckStops	1983	USA	unlimited	unlimited	No fixed limit	User settable, min 1-5 min.	Proprietary heuristics	1001+

* 50 routes, 1,000 stops, two-hour hard-time windows problem

3.3 Matching Algorithm

In this section the research questions (section 2.2) regarding the Matching problem are answered. The research questions "Which type of matching algorithms exists in literature" and "Which marching algorithm performs best for 'Upload!'" contain a knowledge question. These two research questions are translated into three knowledge questions which are shown below and answered in section 3.3.1 through 3.3.3:

1. What is Supply chain collaboration?
2. Which order sharing techniques exists in literature?
3. Which vehicle capacity sharing techniques exists in literature?

3.3.1 What is Supply chain collaboration?

Simatupang and Sridharan (2002) define the collaborative supply chain as "two or more independent companies that work jointly to plan and execute supply chain operations". Collaboration will often lead to better performance compared to operation in isolation. Companies are however only willing to collaborate if doing so contributes to their survival probabilities. Each of the co-operators seeks to achieve individual benefits.

Collaboration within a supply chain can be differentiated in terms of its structure: either vertical, horizontal or lateral. In Figure 3.4 an example of a supply chain is outlined, indicating the different structures of collaboration. Vertical collaboration occurs when different type of actors (e.g. a supplier, a carrier and a retailer) share data and responsibilities. A common example is that actors within a supply chain share demand data to decrease the bullwhip effect. Horizontal collaboration on the other hand is between multiple actors of the same type (e.g. between multiple carriers). Lateral collaboration is collaboration over the whole supply chain, and integrates both horizontal and vertical.

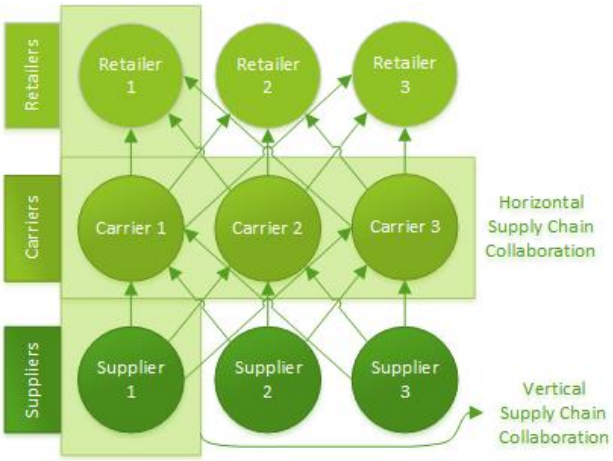


Figure 3.4 Different ways of supply chain collaboration

For the distribution sector, Morash, Droge, and Vickery (1996) have identified that the most important competitive advantages include (i) delivery reliability, (ii) delivery speed, (iii) widespread distribution coverage and (iv) low total cost distribution. Targets (ii) to (iv) can be reached more easily using horizontal supply chain collaboration, whereas the delivery reliability (i) might be a risk when applying horizontal supply chain collaboration since control over the delivery is can be in the hands of another transportation company.

Verdonck, Caris, Ramaekers, and Janssens (2013) identifies two methods for horizontal supply chain collaboration between transportation companies; companies can either collaborate by exchanging orders (Section 3.4.2) or by sharing vehicle capacity (Section 3.4.3). Exchanging orders means that company B delivers an order for company A whereas sharing vehicle capacity means that companies A and B are simultaneously or sequentially using the capacity of a single vehicle.

3.3.2 Which order sharing techniques exists in literature?

Order sharing techniques allow for a proper collaboration between transportation companies to improve the companies' efficiency and profitability. Literature for order sharing techniques mainly focuses on two techniques, joint route planning or auction based mechanisms to share orders. Some other methods are also used, which are discussed under the heading "Other order sharing techniques". In Table 3.3 an overview of the possible solution methods is shown.

Table 3.3 Overview of order sharing techniques, based on (Verdonck et al., 2013)

Order sharing technique	Solution approach	References
Joint route planning	Modified savings heuristic	1
	Local search method	1,2
Auction- based mechanisms	Combinatorial Auctions	3,4
	Iterative combinatorial auctions	5
	Vickrey auctions	6
Bilateral lane exchange	MCLCP: Commercial solver	7
Outlier based exchange	Space-filling curves (OROD) + swap algorithm	8
Shipment dispatching policies	Policy based	9

1 Cruijssen, Bräysy, Dullaert, Fleuren, and Salomon (2007) *2 Krajewska and Kopfer (2006)*
3 Regan and Song (2003) *4 Schwind, Gujo, and Vykoukal (2009)* *5 Dai and Chen (2011)*
6 Berger and Bierwirth (2010) *7 Özener, Ergun, and Savelsbergh (2011)* *8 Clifton et al. (2008)*
9 Zhou, Hui, and Liang (2011)

Joint route planning

Joint route planning basically means that instead of a per route planning, a global route planning for all collaborators simultaneously is executed. This is the most far fetching collaboration method and leads, when solved to optimality, to the shortest distance solution possible. The different solution approaches used are all solution methods for the VRP. The main drawback of this method is that all information is shared between all partners. Even between non-competing companies this is an obstruction, and for competing companies this is a no-go. Real-life examples of joint route planning are found between different divisions and locations of the same company (Krajewska & Kopfer, 2006) and within company-cooperation's. (Cruijssen et al., 2007)

Auction based order sharing mechanisms

Different implementation forms of auction based order sharing have been suggested in literature, but most methods follow the same pattern. Each individual carrier determines which customer requests (orders) are not profitable because of the covered distance. This determination can be done either before the route problem is solved (Regan & Song, 2003) or after the VRP is solved (Schwind et al., 2009). The non-profitable orders are than shared among a group of collaborators, where, by means of some sort of auction mechanism, companies are matched to exchange the orders. In many situations, the bids of the collaborators are compared to a reservation price; if the lowest bid is above the reservation price, the exchange is cancelled.

Both Regan and Song (2003) and Schwind et al. (2009) use combinatorial auctions, meaning that bids on a combination of orders is possible. Reservation prices are calculated based on the difference in total transportation costs including and excluding the combination of orders. Berger and Bierwirth (2010) focuses especially on the amount of information exchanged between collaborating carriers and distinguish between a decentralised approach where minimal information is exchanged and a centralised approach where full information exchange with the authority is required. In the decentralised approach, individual transportation companies seek to maximize their own profit, whereas in the centralised approach an authority seeks to maximise the overall profit.

Other order sharing methods

Özener et al. (2011) uses the method of lane covering to decompose the collaboration problem and so determine the maximum benefit that can be obtained when collaborating. Lane covering is a relaxation of the VRP where freight cannot be combined, e.g. every location requests a full truck load. This decomposition leads to a set of orders to exchange between collaborators. They further elaborate on the amount of information that must be shared between the collaborating transportation companies and simulate the potential benefits for different degrees of information sharing. No real conclusions were drawn from the research, but from the computational study it follows that even when a small amount of data is shared (e.g. only location information of trips with the highest marginal costs), savings up to 40% are possible to derive.

Clifton et al. (2008) uses a space-filling curve mechanism to map delivery locations unto a one-dimensional line (curve). By plotting these locations, the outliers per transportation can be found using a one-dimensional relative outlier detection (OROD) algorithm, and as such determine the potentials for order swapping. Special attention is paid to the information sharing part of the exchange. Using cryptography methods, they assure that no information other than the information which can be deduced from the final swaps is exchanged. They conclude from their study that this method is most useful for full truckload transport.

Zhou et al. (2011) focuses on the situation where certain origin-destination pairs occur more frequently and orders can be collected over a certain time. The truck is than dispatched to the

destination either when the truck is full or the first order must be delivered. They investigate the potential of order sharing either using a strategic alliance, in which case orders from allied companies is picked up when the truck is dispatched while not fully loaded, or full collaboration. In the last case, the companies act as a single company and trucks are dispatched when the combined freight equals a full truckload. From industrial data, it follows that significant improvement can be made in this situation, especially in the case of full collaboration.

3.3.3 Which vehicle capacity sharing techniques exist in literature?

Sharing vehicle capacity means that one truck is used by more than one transportation company. This can either be done using simultaneous operation (orders from different companies in the same truck at the same time) or using sequential operation (capacity of a truck is used by company A first and by company B afterwards). Solution techniques in literature include mathematical programming and negotiation protocols. An overview of the possible vehicle sharing techniques is shown in Table 3.4.

Table 3.4 Overview of vehicle sharing techniques, based on (Verdonck et al., 2013)

Order sharing technique	Solution approach	References
Mathematical Programming	MIP: column generation and decomposition	1
	DDSCCP: branch-and-cut algorithm	2
	Inverse optimisation of flow problem	3
	Rich VRP: decomposition & ant colony opt.	4
Negotiation protocols	Bargaining protocol	5

1 Agarwal and Ergun (2010) *2 Hernández, Peeta, and Kalafatas (2011)* *3 Houghtalen, Ergun, and Sokol (2011)*
4 Sprenger and Mönch (2012) *5 Fischer, Mu¨ller, and Pischel (1996)*

Mathematical programming

Agarwal and Ergun (2010) propose a column generation-based algorithm and a Benders decomposition-based algorithm to horizontally pool liner shipping carrier fleets to operate them together. The capacity is shared among different operators and an optimal solution for all collaborators is determined simultaneously. Since the goal of individual carriers remains to maximise individual profit, a concept from game-theory is used to share the benefits of the method.

Hernández et al. (2011) focuses especially on LTL carrier cooperation for road transportation carriers. The problem is described as a deterministic dynamic single carrier collaboration problem (DDSCCP). Dynamic in this formation means that availability of collaborative capacity is time-dependent. A carrier can acquire capacity from his cooperation partners on their time-dependent availability. The DDSCCP can be mathematically described as an ILP which minimises the total carrier costs. Houghtalen et al. (2011) uses a technique comparable to Hernández et al. (2011), but they formulate the objective to maximise collaborative profit. They state that to maximise collaboration profit, the collaboration partners must have incentive to accept cargo from other carriers and must be encouraged to make decisions profitable for all partners. The model they created is applied in the cargo airline industry.

Sprenger and Mönch (2012) uses a somewhat different approach where the capacity sharing technique is integrated within the VRP formulation. The VRP for all the collaboration partners is decomposed based on network zones. Each network zone corresponds with one of the collaborating partners and capacity is shared for orders from company A located in the network zone of company B. They tested their algorithm on real-world data from food manufacturing companies in Germany.

They conclude that capacity sharing leads to a reduction in the travelled kilometres and the number of time-window violations.

Negotiation protocols

Fischer et al. (1996) propose a method that forms the fundamentals of the MARS (Modelling Autonomous coopeRating Shipping) environment. MARS is used to model for a group of shipping companies the orders in a cooperative way. Each truck functions as an agent which negotiates on each incoming order to obtain an optimal schedule for each of the trucks, independent of the company which executes the order. This leads to decentralised planning since every individual truck remains owned by its original owner, no centralised planning entity is used.

The individual truck agents use an auction protocol to globally determine the best truck to allocate the order to. Trucks owned by the order owner are favoured over other trucks to limit the amount of cooperation and information sharing. The implementation of this model also foresees in situations where a certain order cannot be fulfilled due to dynamic situations such as traffic jams. A new negotiation can start to make use of unused capacity of vehicles of cooperating companies to still can fulfil orders.

3.3.4 Auction mechanisms

Following McAfee's (1987) definition of an auction: "a market institution with an explicit set of rules determining resource allocation and prices based on bids from the market participants", it becomes clear that for each auction there must exist a set of rules determining when and how a deal between seller and buyer is made. There are two main criteria indicating a good set of rules, first the allocative efficiency; does the object end up at the bidder who values the objects the most and secondly the revenue, maximising the expected selling price. (Bichler, 2010)

Auctions can either involve one object (single unique indivisible object), multiple objects of the same type (single unique divisible object) or multiple objects of different types, either in multiple of single quantities (multi (in) divisible objects). These four types of auctions require a different set of rules, because of the difference in the nature of the auction. Other characteristics of an auction include if the bidders or their agents are physically present at the location of the auction, whether there is a reservation price, that is, a minimum bid required to obtain the item (if no bid exceeds the reservation price, the item is not sold) and a pre-set buy-out price, indicating that at any time a bidder can pay the buy-out price and thereby removing the item from the auction.

These auction characteristics are often determined by the nature of the auction. Besides that, there are different types of auction mechanisms, distinguishing in the way the price of an item is determined. There is a distinction between single and multiple bids per bidder, increasing or decreasing bids and the way the price of the item is determined after the bids are placed. To illustrate this some well-known examples for unique object auctions are shown in Table 3.5.

The **English auction** is probably the most common form of auction {Krishna, 2002} and uses multiple bids where each subsequent bid must be higher than the previous bid. When no participants are willing to bid higher, the item is sold to the highest bidder. Although the highest bid is paid, this is not necessary the amount the highest bidder values the object, but it's the amount the second highest bidder values the object, plus an arbitrary additional amount to surpass the second highest bid.

Table 3.5 Overview of single item auctions

	Object	Bids per participant	Price determination	Examples
English auction ***Open ascending price auction***	Indivisible	Multiple	Highest bid	Antique Real estate
Dutch auction ***Open descending price auction***	Divisible	Single / Multiple	First bid	Perishable commodities
Blind auction ***First-price sealed-bid auction***	Indivisible	Single	Highest bid	Company procurement
Vickrey auction ***Second-price sealed-bid auction***	Indivisible	Single	Second highest bid	Stamps Agent biddings

The **Dutch auction** varies from the English auction in that the price is starting high and lowered as the auction continues; where at the moment the first bidder stops the auction this bidder can buy a certain amount of the object auctioned. The auction then continues until either all is sold or the reserve price of the object is reached. Multiple bids per participant are possible, first securing a required amount of the object, and later buy an additional amount for a lower price.

In **blind auctions,** every bidder placed simultaneously one bid to the object; whereby the participants do not know each other's bids. The highest bidder then pays their bid price. Disadvantage of this method is that there is no price discovery of the item. For example, in an English auction information about the preferences of other participants can follow from earlier biddings. When the value of an object is unknown, blind auctions might lead to lower revenue for the auctioneer.

The **Vickrey auction** is identical to the blind auction with the exception for the price determination. The highest bid wins the auction, but the price equals the bid of the second highest bid. This encourages truthful biddings, since it can be shown mathematically there is no incentive to strategically bid different then the price the participant values the item.

For multi-item auctions often modification of these auction methods are used.

3.4 Conclusion

In this chapter, relevant literature regarding the Vehicle Routing Problem and the Matching problem is discussed. The results of this overview are used in the following section where we determine a possible solution method for the VRP and the MP.

4 SOLUTION APPROACH

In this chapter the translation between the literature and the implementation of the algorithms is made. The motivation for the selected methods is described and the implementation methodology is explained. The results of the implementation steps can be found in the next chapter.

4.1 VEHICLE ROUTING PROBLEM

In this section the methodology for the implementation of the VRP model is described. The choice for the used algorithms is explained and the algorithms are described.

4.1.1 Classification of problem

In the literature review (Figure 3.1 in Section 3.2.2), a framework of constraints, features and conditions applicable for the VRP, based on the work of Ganesh et al. (2007) is shown. In Figure 4.1 we show which constraints are required for 'Upload!' given the context analysis in Section 2.1. We divide these constraints into 3 categories, the main constraints that are valid for most potential participants of 'Upload!', the potential constraints which are not directly mentioned in the profile sketch but are likely and finally the possible constraints which will be valid only for a small portion of the participants of 'Upload!'.

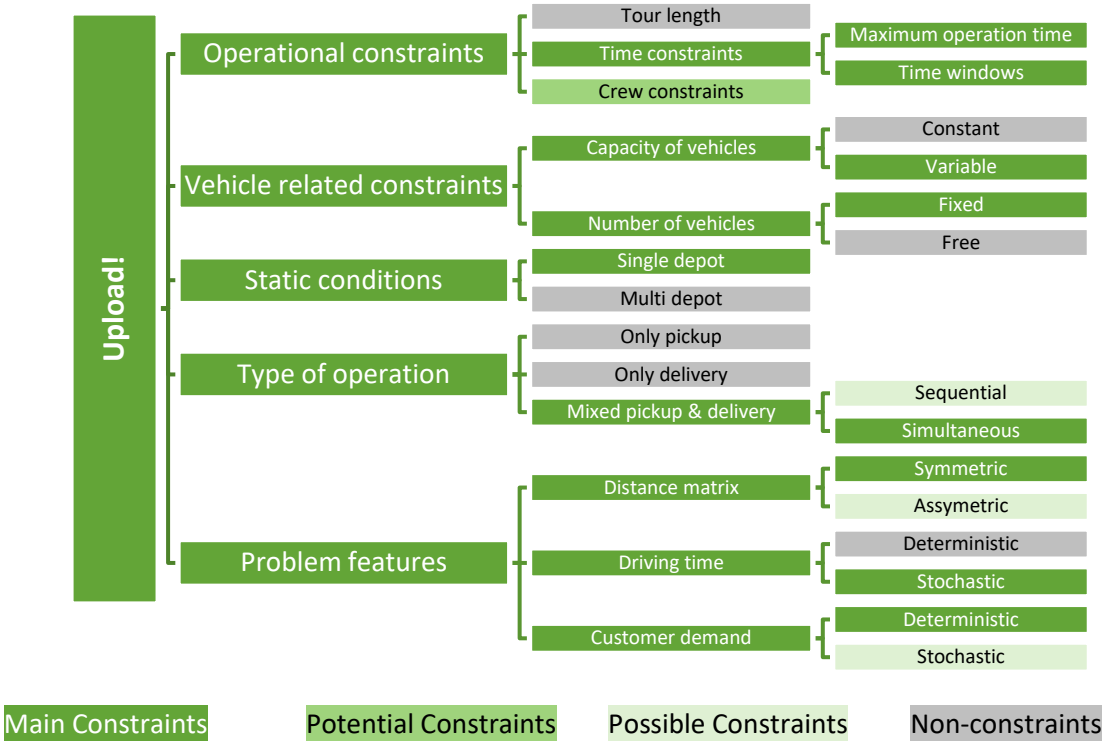


Figure 4.1 Framework showing the different constraints for 'Upload!' route planning

For the operational constraints, the tour length is usually not a constraint for the smaller transportation companies. Due to government regulations however, there are constraints on maximum operation time. Time windows (either hard or soft) will always be used for transportation companies carrying load for other companies and finally crew constraints might occur when there are multiple drivers.

As for the vehicle related constraints, we have seen that for these transportation companies the number of vehicles is fixed and the capacity of the different vehicles (for companies with more than one vehicle) might vary. As for the number of depots, smaller companies will usually use only one single depot.

The type of operation is mixed pickup and delivery, since no freight is carried from or to the depot. This pickup and delivery will usually be simultaneous; although there might be situations in city delivery where first all freight is collected outside the city centre and then distributed within the city. As for the problem features, distances are mostly symmetric (except for some one lane city streets), driving times are stochastic due to the time influences (rush hours) and the customer demand will mainly be deterministic, but might occasionally be stochastic.

4.1.2 Selection of constraints to be implemented in proof of concept

In this Section, we determine which of the constraints for 'Upload!' shown in Section 4.1.1. we will implement in the proof of concept. On one hand we want as many constraints as possible, but on the other hand we are limited in implementation time. Furthermore, the available data is limited to benchmark instances on an x-y plane since there are no participants in 'Upload!' just yet. Overall, we choose the following simplifications:

- Maximum (consecutive) operation time is not implemented since this complicates the VRP by a lot. Furthermore, this is also not enforced in any benchmark and therefore makes benchmarking difficult. An estimation of the influence of maximum operation time can be made by multiplying the travel times by a constant indicating the percentage trip breaks during a trip (if for instance the maximum consecutive driving time is two hours, and after that a break of a quarter of an hour is required, the trip duration can be multiplied by 9/8). This additional travel time compensates for the required breaks.
- Crew constraints are not implemented since not much is known about this in advanced. Company specific rules must be followed, and therefore company specific adjustments to the algorithm must be made.
- For the vehicles we choose for a slightly different approach, where the number of vehicles is free and the optimisation goal is to minimise the number of vehicles used. As such we both maximise the utility per vehicle and on the other hand may discover the need for additional vehicles which is best for the exchange step.
- Sequential insertion for one central point for city delivery can be enforced by adjusting the time-windows at this pickup point and therefore is not implemented explicitly.
- Finally, and maybe most importantly we choose to implement a static version of the VRP where customer demand is known upfront and no additional orders are added during the execution. These exceptions can first be handled using a management by exception method and implemented as a feature during a later stage.

By applying these simplifications, the constraint framework shown in Figure 4.1 changes to that of Figure 4.2. In the remainder of this thesis we will elaborate upon these constraints. In Section 4.1.3 a mathematical description of the problem is provided and starting in 4.1.4 we describe the selection and implementation of the models used to solve this VRP.

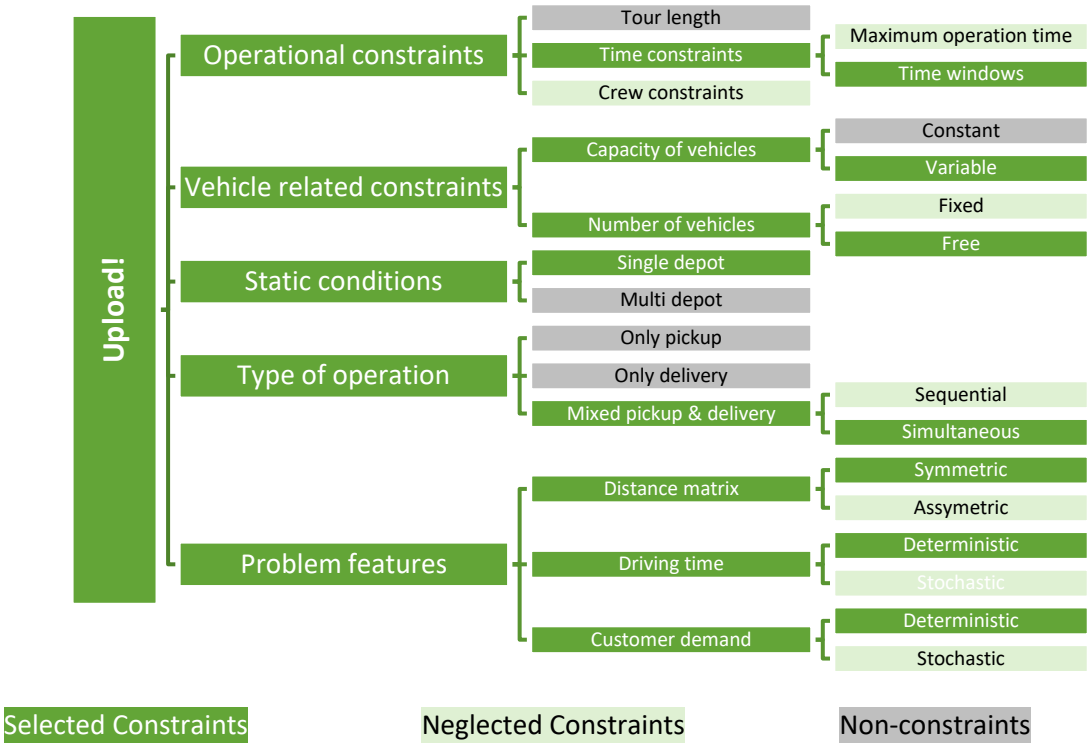


Figure 4.2 Framework showing the selected constraints that are going to be implemented in the proof of concept

4.1.3 Problem formulation

To describe the VRP we use a pseudo mathematical formation. Because we do not intent to use an exact solution method an ILP formation of the problem is not required. A clear overview of the problem with the constraints however does help to clarify the problem and understand the proposed solution.

$$\min z = c_1 \cdot \#vehicles + c_2 \cdot distance + c_3 \cdot time \quad (4.1)$$

Subject to:

- All order points must be visited exactly once
- The full order quantity must be loaded or unloaded at the order point
- Every trip *t* must start and end at the depot
- All order points *i* must be visited within their time window $[a_i, b_i]$
- Arrival time at order point *n+1* = departure at order point *n* + travel time between *n* and *n+1*
- Departure time at order point *i* = max(*arrivaltime*$_i$, a_i) + h
- The used capacity of the truck may not exceed the maximum capacity
- For every order, the pickup location must be visited before the destination location and both must be visited by the same truck.

The objective is to minimise the sum of total travel distance, travel time and number of vehicles, each multiplied by a positive coefficient to be able to give the optimisation required by the company. The constraints are directly related to the characteristics of VRPs in general and the classification of the VRP shown in Section 4.1.1.

4.1.4 Choice of model

Various models described in the literature review can be used to solve the problem formulated in Section 4.1.3. Neighbourhood searches have the advantage that they generate solutions better solutions depending on the runtime of the algorithm, which means that runtime can be a parameter in the optimisation. Furthermore, neighbourhood searches provide a general solution method which can be used in a broader range of similar optimisation problems. Of the two most used neighbourhood search algorithms (Simulated annealing and Tabu Search), we selected simulated annealing as model because it is easier to incorporate different neighbourhood construction methods and it is in our eyes also easier to implement.

Any neighbourhood search, and especially if time is bound, profits from a so called warm start. This means that the start solution is already a relative good solution to the problem. We constructed a variant of savings algorithm adjusted to incorporate pickup and delivery to create this warm start. The savings algorithm has as main advantages that it is both quick to implement and quick to execute. A description of each of these methods is shown below.

4.1.5 Model description Savings algorithm

Clarke et al. (1967) were the first to introduce the concept of savings as a heuristic to solve the VRP. A savings algorithm starts with the situation where every order is delivered by a separate vehicle. When two orders are combined, the total distance will decrease (from the situation where the edges depot – 1 and depot – 2 are travelled twice, now the edges depot – 1, 1 – 2 and depot – 2 are travelled). When assuming the triangle inequality, the new situation leads to the same or a smaller distance. Some modifications have been applied to make the basic savings algorithm suitable for the problem of 'Upload!' In Figure 4.3 a flowchart regarding the modified savings algorithm is shown.

Initialisation

In the initialisation phase, we start with reading the start situation (list of all orders, vehicles, parameter settings and location of the depot. The distance and travel time matrices are calculated or read from input and a trip is created for every order.

From this start solution where every order (pickup and delivery combination) is visited in one single trip, a savings list is created by calculating the decrease in the objective value when merging two trips. The merging of trips is done based on the method shown in Figure 4.4. Instead of calculating the solution value for every possible order configuration (which is $O(n!)$ in the number of possible configuration, if n denotes the number of order points in both trips), we use a sequential insertion method. The sequence of trip 1 is kept intact and every order of trip 2 is added sequential, where each order point remains its order relative to the previous order point in the trip (reducing the number of solution values to evaluate to $O(n)$). Only valid savings (savings that lead to trips without time window or vehicle capacity violations) are added to the list.

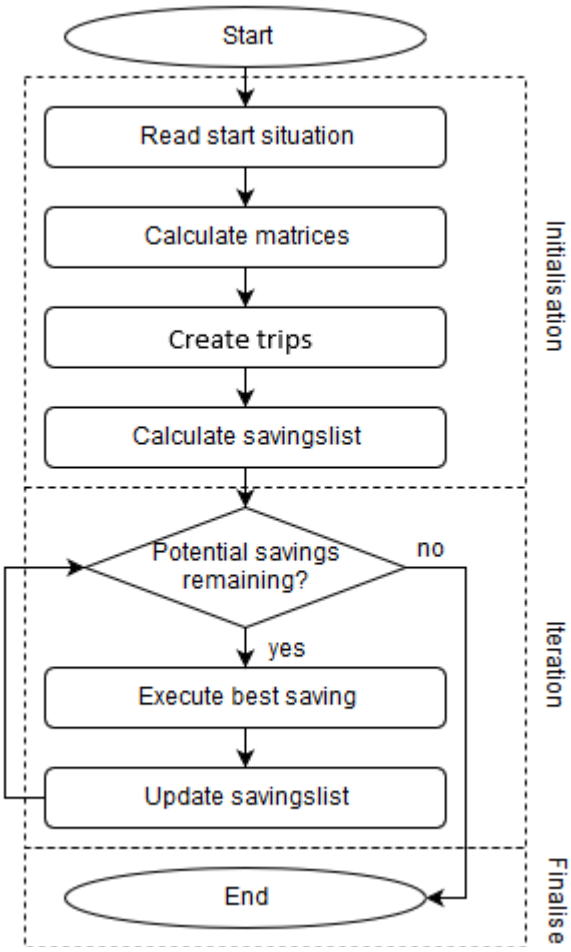


Figure 4.3 Flowchart of Savings Algorithm

Iteration

While there are remaining potential savings on the savings list, the saving with the highest potential is selected and executed, after which the savings list is updated. Executing the saving means that the virtual vehicles involved are removed and one new virtual vehicle containing the combined orders is added. Updating the savings list means removing all saving with the involved virtual vehicles from the list and adding every potential combination involving the newly created vehicle.

Finalisation

The result of this algorithm is a limited set of trips. Advantage of this method is that it gives a warm start for the second part of the VRP solver, the local search algorithm. The results of the savings algorithm for various benchmark instances are shown in Section 5.2 and discussed in section 6.2.1.

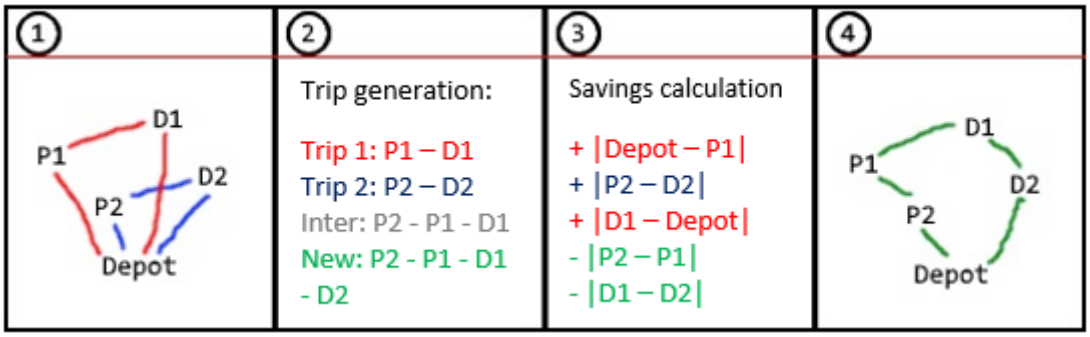


Figure 4.4 Detail flowchart of process of merging two vehicles

4.1.6 Model description Local Search algorithm

As motivated in Section 4.1.4, simulated annealing is used as a local search algorithm. Simulated Annealing is based on the process of annealing in metallurgy (hence the name) and uses the concept of a decreasing temperature to limit the acceptance ratio of non-optimal modifications over time. In every step a new solution (neighbour solution) is generated by a small modification in the solution. When the temperature is high, many neighbour solutions are accepted, whereas as the temperature decreases, non-optimal modifications are less often accepted. In this way, the method can escape from local minima while in the meantime still determine a good solution. The implementation of simulated annealing is unmodified from the version described in the literature review. First we will describe the process in general and then we will describe the neighbourhood creation method. Finally, we will provide an overview of the parameters used in the method.

Process description

Starting with the solution generated in the savings algorithm, the local search algorithm tries to find better solutions. In Figure 4.5 an overview of the simulated annealing process is shown. The temperature c is initiated to a start temperature c_{start} and is decreased every cycle with a fraction α until a stop temperature c_{stop} is reached. In every cycle a number of iterations k (also known as the Markov chain length) is performed. The "generate neighbour" procedure is the local search component of simulated annealing. Based on the current solution, a small modification using one of the three methods shown under the heading "Neighbourhood operators" below is performed. The new solution, called a neighbour solution is either accepted as the new current solution, or is rejected based on Formula (4.2).

$$Accept\ if: e^{\frac{\Delta}{c}} = e^{\frac{|current|-|neighbour|}{c}} > Rand(0,1) \qquad (4.2)$$

Here Δ denotes the difference between the current solution value and the solution value of the neighbour. If the neighbour is better, i.e. has a lower solution value, it is always accepted since the exponent of something above 0 is always greater than 1. If the solution value is smaller than the best so far, it replaces the best solution. This iteration is continued while the current temperature is above the stop temperature.

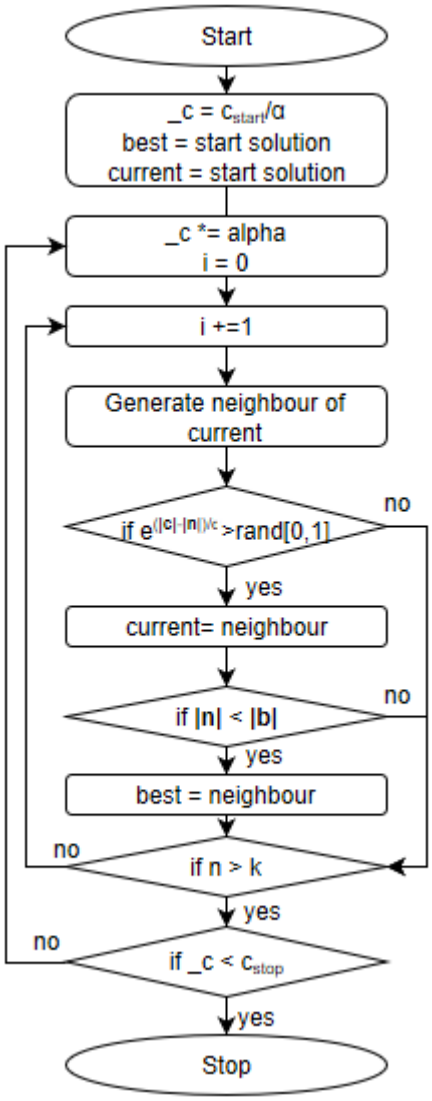


Figure 4.5 Process of Simulated Annealing

Neighbourhood operators

A neighbourhood solution is a solution that is like an existing solution, but with a small modification. This modification is created by an operator on the existing solution. Aim is to use operators which do small modifications yet are covering the whole solution space when applied consecutive. Therefore, the total set of operators must be able to create additional trips, delete existing trips and change the sequence in which the order points within a trip are visited. The route improvement heuristics introduced in Section 3.2.3 are an example of such operators.

The most basic operation is to move one order from one trip to another one. This gives the possibility for a trip to be removed (if there are no orders in the trip remaining), and for a new trip to be created (by moving to an empty trip). It can also reposition order points within a trip (by first removing the order points from a trip and then later reinserting it back at a different location), but this is more consuming. To make it easier to move an order point within a trip another method is used, the pointmove operator. Another disadvantage of the ordermove operator is that it may not be feasible to move an order from one trip to another trip before first "making space" for the receiving order. This is intercepted by introducing a third operator, the orderswap operator. This operator exchanges two orders between two trips, e.g., it removes two orders from two different trips and then replaces these orders in the other trip.

A description of how the operators are implemented is shown in the enumeration below. The operators are placed in the order of time consumed by each operator.

1. **Pointmove**; One order point (either pickup or delivery) is randomly selected from a random trip and is moved within that trip to another position, obeying the precedence relations between pickup and delivery.
2. **Ordermove**; One order (pickup and delivery point) is randomly selected from a random trip and is swapped to another randomly selected trip. The order is then inserted on the best available position on that trip (minimising the objective including constraints). This operating also includes the possibility to delete trips from the solution (by removing the last remaining order from a trip) and to add trips to the solution (because the random trip where the order is inserted can also be an empty trip.
3. **Orderswap**; Like the procedure of the ordermove, although here two random orders from two random selected trips are removed and reinserted in the other trip.

Each of these operators is selected at random given a probability p_1 through p_3 for each of these methods. In principle, only method 2 allows for all possible neighbourhoods, and will therefore probably have the highest probability. The other two methods however allow for respectively quick rearrangements within a trip (without the pointmove operator the only way to modify the sequence within a trip is by first removing the order (ordermove 1) from the sequence and later reinserting it in the same trip (ordermove 2), which has a small probability) and for more options once the trips are all about equally filled, the orderswap operator is introduced, which is easy to implement and can make bigger steps at once.

Parameters

The simulated annealing parameters ($c_{start}, c_{stop}, k, \alpha$) and the parameters to determine with which probability one of the three methods to construct a neighbour solution is chosen, are determined in Chapter 5. Also in this chapter the quality of this VRP solution method is determined by comparison with existing benchmark data.

Large Neighbourhood search

Based on the literature review, we propose to use Large Neighbourhood search as an additional neighbourhood operator. Large Neighbourhood Search (LNS) is a method that can quickly move from one solution to a whole new type of solution by destructing a given percentage of the solution and afterwards rebuilding the solution. It uses a variable df that indicates the percentage of the solution that is destructed. An order is removed from the solution if a random number between 0 and 1 is smaller than or equal to the destruction factor df. The result is a set of trips with empty space in it and a list of orders that must be reinserted in the trips. All the orders that have been removed from the solution are then sequentially reinserted in the best available trip at the best available location within that trip. The order in which these orders are reinserted is also at random.

4.2 SELECTION OF INEFFICIENT TRIPS

The found literature regarding the creation of the decision tree (the selection of the orders that are offered on the marketplace) was very minimal and is integrated in the section about the matching algorithm. To come up with a set of criteria, a stakeholder analysis was performed and the expert opinion of Infostructure was used. This eventually led to the selection criteria shown in section 4.2.3.

4.2.1 Stakeholder Analysis

For a typical transportation company, the stakeholders are as shown in Table 4.1. The classification of the stakeholders was done based on the saliences introduced by (Mitchell, Agle, & Wood, 1997). We see that from inside the company the truck drivers and the freight planners are important, whereas from outside of the company the shippers are the most important stakeholders.

For each of these stakeholders their typical goals or targets relating to the routing problem are identified, see Table 4.2. Most of these goals have strong overlap with each other; e.g. the minimisation of the driven distance and the minimisation of CO_2, and as such the not many trade-offs have to be made. A Trade-off must be made between the time-windows and the minimisation of transportation costs, but these are solved by either adapting a strategy of hard time-windows or associating a cost to lateness. Regional constraints on one hand can strive with the flexibility of the VRP solving and therefore introduce higher transportation costs, but will on the other hand lead to better punctuality of the planned route and is therefore also not a real trade-off.

Table 4.1 Overview and classification of stakeholders, based on their salience, based on (Mitchell et al., 1997)

	Power	Urgency	Legitimacy	Type of stakeholder
Transportation company management	V		V	Dominant stakeholder
Truck drivers		V	V	Dependent stakeholder
Freight planners		V	V	Dependent stakeholder
Shippers	V	V		Dangerous stakeholder
Receivers		V		Demanding stakeholder
Environment		V		Demanding stakeholder
Law	V			Dormant stakeholder

Table 4.2 Typical goals of identified stakeholders

	Targets

Transportation company management	• Minimise transportation costs
	  ○ Maximise truck utility
	  ○ Minimise driven distance
	  ○ Minimise # of used vehicles
	• Minimise shared data between companies
	• Maximise customer satisfaction

Truck driver	• Minimise on trip waiting time • Maximise use of regional constraints
Freight planner	• Maximise simplicity of planning
Shipper	• Pickup package within time-window • Deliver package within time-window
Receiver	• Receive package within time-window
Environment	• Minimise CO_2 emission
Law	• Routing in accordance with the social legislation on road transport {EG561, 2006}

Based on these goals, the selection criteria for inefficient trips and (combination of) orders, which are mentioned in Section 4.2.2 and 4.2.3 respectively, are constructed. Based on these inefficient, sets of inefficient orders are determined.

4.2.2 Performance of trips

Following the results of Section 4.2.1., we see that the stakeholders generally want trips that are in accordance with the time windows, which is already a restriction in the final result (because of comparison with benchmark). Furthermore, the drivers want to minimise on trip waiting time, which is in alignment. Overall, we came up with one measure of the financial performance of the trip and one combination of performance indicators describing the trips itself.

For the financial performance of the trip we compared the travel costs (a combination of fixed costs per truck and a price per kilometre and per unit of time) with the expected revenue. The revenue is a function of the distance the pickup and delivery location are apart from each other and the amount transported. For convenience we state that in this research the gain is a fixed amount per unit distance per unit load transported. The exact value of this gain is not important since we look at the

ratio between revenue and costs, which scales by the exact value, but does not influence the comparison between different trips. In Formula 4.3 the calculation is shown.

$$PI_j^f = \frac{revenue\ of\ trip\ j}{costs\ of\ trip\ j} = \frac{\sum_o^{O_j} dist\left(o_{pickup}, O_{delivery}\right) \cdot o_{quantity}}{c_1 + c_2 \cdot distance_j + c_3 \cdot time_j} \tag{4.3}$$

The values c_1 through c_3 here represent the same values as in the minimization objective stated in Section 4.1.3. The set O_j denotes all trips o executed by truck j. The function dist() calculates the distance between the pickup and delivery position of order o and finally the subscript quantity denotes the order quantity associated with the truck. The result PI^f_j is a measure how well the trip performs finance wise. The higher this value, the better the trip performs (higher ratio of revenue to costs).

Besides a financial measure, we also introduced a performance measure that can be more directly related to the resulting trip, without having to look at the order details of the orders transported by this truck. We use a combination of load factor, waiting time, orders per trip and relative distance. This is the so called empirical performance indicator of the trip, and the way it is calculated is shown in Formula (4.4)

$$PI_j^e = w_1 \cdot \frac{|J| \cdot loadfactor_j}{\sum_k^J loadfactor_k} + w_2 \cdot \frac{|J| \cdot waiting_j}{\sum_k^J waiting_k} + w_3 \cdot \frac{|J| \cdot |O_j|}{\sum_k^J |O_k|} + w_4 \cdot \frac{distance_j}{\sum_o^{O_j} dist\left(o_{pickup}, O_{delivery}\right)} \tag{4.4}$$

$$loadfactor_j = \frac{\sum_e^{E_j - 1} dist(e, e+1) \cdot L_e}{Cap_j \cdot \sum_e^{E_j - 1} dist(e, e+1)} \tag{4.4a}$$

$$waiting_j = \frac{\sum_{n=1}^{N_j} \max(A_n - arr_n, 0)}{arr_0 - dep_0} \tag{4.4b}$$

$$distance_j = dist(o_0, n_1) + \sum_{n=1}^{N_j - 1} \mathrm{dist}\,(\mathrm{n_i}, \mathrm{n_{i+1}}) + dist\left(n_{N_j}, o_0\right) \tag{4.4c}$$

The values associated with the load factor and the number of orders per trip (w_1, w_3) will be positive, since high load factors and a high number of orders per trip are preferable. The values associated with waiting time and relative distance (w_2 and w_4) on the other hand will be negative, since waiting time and long relative distances are undesirable. In Chapter 5, we compare this two performance measures and determine what is best way to determine inefficient trips.

4.2.3 Selection criteria for inefficient trips

Although the performance of the trips can now be quantified, this is not yet a selection criterion. A cut-off value must be determined below which a trip is considered inefficient. This cut-off value will vary from company to company since performance in one sector might be considered good while in more competitive sectors is might be considered inefficient performance. After consideration with the Infostructure we decided to use two criteria, one based on the absolute value of the performance, which can only be quantified after a longer initialisation time, and one based on the relative value compared to other trips of that day. This leads to a selection criteria which both

include the bottom performing trips per day and the bottom performing trips overall. In Formula (4.5) this relation is shown.

$$Q_j = \begin{cases} 1 & | \quad PI_j^e < t_1 \; OR \; PI_j^e < t_2 \cdot \frac{\sum_{d=1}^{J} PI_j^e}{|J|} \; OR \; PI_j^f < t_3 \; OR \; t_4 \cdot \frac{\sum_{d=1}^{J} PI_j^f}{|J|} \\ 0 & | \quad else \end{cases} \tag{4.5}$$

4.2.4 Performance indicators and selection criteria for inefficient orders

From inefficient trips to inefficient orders is only a small step. The same performance measures are used to determine for any given set of orders, up to a maximum of N orders per trip, which set of orders are responsible for the bad performance of the trip. The formulas are similar to those of the selection criteria for inefficient trips, although here for the empirical measure the values are related to the total of the trip instead of to the average over all trips.

Then once again the step from performance indicators to selection criteria must be made. To determine which combination of orders cause the inefficiency of the trip, the trip performance with and without these orders is compared. The highest improvements are selected.

4.3 Matching Algorithm

In the last step of the algorithm, the actual exchange between different participants of 'Upload!' takes place. In this Section the algorithm choice (Section 4.3.1), and the working of the algorithm (Section 4.3.2) is explained. In Section 4.3.3 a small example of the working of the algorithm is shown.

4.3.1 Choice of algorithm

To determine which algorithm to use for the exchange of orders between 'Upload!' participants, we compared different auction mechanisms based on whether these methods are:

1. Combinatorial auction: The participants must have the possibility to bid at multiple items simultaneously and may accept a certain order only if also a second order is acquired.
2. Truthful bidding is optimal: The participants must be ensured that bidding the amount they actually value the object is the optimal bidding strategy.
3. Single bid: Every participant gives a single bid for every unique combination of orders available.

The auction mechanism that was designed for this situation is the Vickrey-Clarke-Groves (VCG) auction. The working of the VCG auction is described in the next section.

4.3.2 Working of Vickrey-Clarke-Groves auction

A flowchart of the VCG auction mechanism is shown in Figure 4.6. The process starts with creating all possible combinations of auctions. We choose to use binary representation to indicate which order combination includes which order. The order combination 0110 for example contains the orders 2 and 3 (counting from the right, the index of order 1 = 0, of order 2 = 1 etc. In the next step every participants determines a bid price for

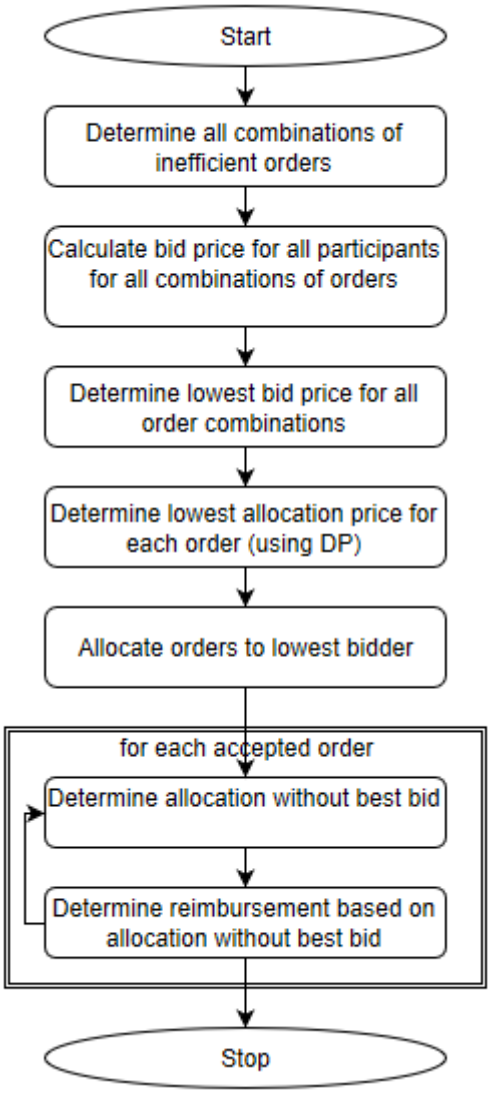


Figure 4.6 Flowchart of the VCG auction mechanism

each order combination. This is done by taking the insertion price of an order in the existing solution using the same method as the insertion of orders in the ordermove operator (Section 4.1.5). We limited insertion of order combinations to insertions in one trip, since there is no additional benefit for the bidder to bid on a combination of orders if these orders are inserted in different trips. Instead the bidder places a bid at the individual items (or combinations) which are placed in different trips. The owner of the order also places a bid on these orders. This ensures that the reimbursement the order owner has to pay never exceeds the amount it has to pay to perform the delivery himself.

Once all bid prices are known per order combination the best bid price and corresponding bidder is selected. Using dynamic programming (see the example in Section 4.3.3) the best (lowest) bids per order are determined and are allocated to its bidder. Dynamical programming is an exact solution method to this NP-hard allocation problem. Before allocating the orders to the best bidders, we check for one additional constraint. Either no order returns to the original owner, or all orders of a set from one inefficient trip return to the original owner, since the total execution costs may never increase for the owner of the orders.

We than know which order to allocate to which participant. We do not yet know the reimbursement because if the bid price is used, the system is susceptible for untruthful valuation of the orders. To determine the amount the bidder has to pay we determine the bid price if this participant would not have participated. This simulates the effect of multiple biddings where the highest bidder also just has to pay the amount the second highest bidder valuates the project (plus a slight amount to overtake the second highest bidder). In order to determine the result of the auction without this bidder active we temporarily remove this bidder from the results. We again determine the best bid prices and the best allocation. The reimbursement will then be equal to the bid price in this situation. This procedure is repeated for every participant who wins a not already owned order.

4.3.3 Vickrey-Clarke-Groves auction example

To explain the principle of the VCG auction and to show how the dynamic programming works we show an example with 4 participants and 3 orders. Suppose an inefficient trip consists of three orders, o_1 trough o_3. There are three bidders, p_1 trough p_3 who ask for a reimbursement so that they can deliver a combination of order lines o_1 through o_3. The original owner of the orders (p_0) also bids, so that the maximum reimbursement is controlled by the order owner. The biddings are shown in Table 4.3.

Table 4.3. Bid prices per participant for all order combinations

	{o_1}	**{o_2}**	**{o_3}**	**{o_1,o_2}**	**{o_1,o_3}**	**{o_2,o_3}**	**{o_1,o_2,o_3}**
p_0	50	75	50	110	80	110	150
p_1	30	70	20	90	40	80	100
p_2	70	60	25	100	75	65	130
p_3	10	120	110	105	100	200	205

As one can see, a combination of orders will always have a bid price less than or equal to the sum of the individual orders. Following the protocol in Figure 4.7, we now determine the lowest bid price for all order combinations (Table 4.4).

Table 4.4. Minimum bid price for each order combination

	{o_1}	**{o_2}**	**{o_3}**	**{o_1,o_2}**	**{o_1,o_3}**	**{o_2,o_3}**	**{o_1,o_2,o_3}**
min	10	60	20	90	40	65	100

The next step is to determine the optimal allocation of order over the participants. In Table 4.5 the recursion of dynamic programming is shown. At a given state, the minimum bid price at that state can be calculated as the minimum of all the combinations that lead to the solution at that state. The state is noted in binary to indicate which states are required.

Table 4.5 Calculation of cost at every state of the DP algorithm

State	Calculation	Price	Allocation
111	$V_{123} = \min(V_{12} + p_3, V_1 + p_{23}, V_2 + p_{13}, V_0 + p_{123})$	$\min(70 + 20, 10 + 65, 60 + 20, 100) = 75$	O_1+O_{23}
011	$V_{12} = \min(V_1 + p_2, V_0 + p_{12})$	$\min(10 + 60, 0 + 90) = 70$	$O_1 + O_2$
010	$V_2 = V_0 + o_2$	$60 + 0 = 0$	O_2
001	$V_1 = V_0 + o_1$	$10 + 0 = 0$	O_1
000	$V_0 = 0$	0	-

The cheapest way to fulfil all three orders is by granting o_1 to bidder 3 and the combination of o_2 and o_3 to bidder 2, at a total cost of 75. However, since we want the bidders to have the incentive to bid their true valuations, the reimbursement price is determined by calculating the price if this bidder would not have bid. If bidder 2 would not have bid, than the best biddings would have been (Table 4.6):

Table 4.6 Best bidding excluding participant 2

	{o_1}	**{o_2}**	**{o_3}**	**{o_1,o_2}**	**{o_1,o_3}**	**{o_2,o_3}**	**{o_1,o_2,o_3}**
min	10	70	20	90	40	80	100

The best bid for a combination of orders 2 and 3 equals 80 and therefore 80 is the reimbursement participant 0 has to pay to participant 2. The same holds when bidder 3 would not have bid. The minimum prices then would have been (Table 4.7).

Table 4.7 Best biddings excluding bidder 3

	{o_1}	**{o_2}**	**{o_3}**	**{o_1,o_2}**	**{o_1,o_3}**	**{o_2,o_3}**	**{o_1,o_2,o_3}**
min	30	60	20	90	40	65	100

Now the best bid for orders 1 equals 30 and therefore 30 is the reimbursement participant 0 has to pay to participant 3. The total reimbursement for participant 0 equals 80+30 = 110, which is a gain of 40 compared to the initial execution costs. The gain for participant 2 and 3 is the difference between the bid price and the actual reimbursement, which equals 5 for participant 2 and 15 for participant 1. Overall the gain of all participants is 60 (40+5+15), which is considered the total gain due to the VCG auction.

4.4 Conclusion

In this section we saw that we can solve the Vehicle Routing Problem using a combination of a Savings Algorithm and a Simulated Annealing algorithm. We described which parameters are required and how we can modify the algorithm to possibly perform even better. Then we discussed the method to determine which trips and orders are inefficient by introducing two performance measures, a financial PI and an empirical PI. Finally we determined to exchange orders between 'Upload!' participants using the Vickrey-Clarke-Groves auction. We described how to determine the bid price for each participant and how to calculate the reimbursement price.

5 NUMERICAL EXPERIMENTS

In this chapter the experimental setup and the results of the experiments are shown. Aim of the chapter is to show how the algorithms proposed in Chapter 4 perform. The chapter is divided into two sections; Section 5.1 describes the experimental setup and Section 5.2 describes the results. Each of these two sections is further divided into three subsections, describing the vehicle routing problem, the selection of inefficient trips and orders, and the matching algorithm. At the end of the chapter (Section 5.3) the conclusion of this chapter are drawn.

5.1 EXPERIMENTAL SETUP

In this section the setup of the experiments is given. The various experiments are explained and both the aim and expectation of each experiment are described. This section is divided a subsections about the benchmark (Section 5.1.1), the Savings algorithm (5.1.2), the Local Search algorithm (5.1.3), the modified Local Search algorithm (5.1.4), the selection of inefficient trips and orders, (5.1.5) and finally the Vickrey-Clarke-Groves auction (5.1.6).

5.1.1 Li Lim benchmark

The Li Lim benchmark was already described in Section 2.3, here we repeat the most important information for reading simplicity. The Li Lim benchmark is a modification of the Solomon instances for Vehicle Routing Problems with time windows, which includes the pickup and delivery aspect. Aim of the benchmark is to a) minimise the number of vehicles needed to perform all pickup and deliveries and b) minimise the total required distance to do so. It consists of six types of instances, varying in order point distribution (order points are distributed over the map either clustered, at random or at a mixture of these two) and vehicle capacity. As shown in Section 2.3, three of these six types of instances are considered relevant for the customers targeted by 'Upload!'. A type of instance is called an instance group in the remainder of this thesis.

In Table 5.1 the characteristics of these three instance relevant groups are shown, including the number of instances in the group, the order point distribution, the vehicle capacity, the handling time, and the average number of vehicles required, average distance and average travel time of the best-known solution. The dimensions of distance and time are units since distance and travel time are equal in the definition of the benchmark. Time and distance therefore only differ due to handling time and waiting time due to time windows.

Table 5.1 Type and number of instances in Li & Lim's PDVRPTW Benchmark

Type	# of instances	Order point layout	Vehicle capacity	Handling time	AVG # of vehicles	AVG distance (units)	AVG time (units)
LC2*	8	Clustered	700	90	3	589.86	9589.86
LRC1*	8	Mixed	200	10	11.5	1386.74	2411.48
LRC2*	8	Mixed	1000	10	3.25	1133.12	2444.54

5.1.2 Savings algorithm

The savings algorithm is the first part of the solution method for the VRP, and is used to provide a warm start, e.g. a good initial start, for the local search algorithm. To follow the benchmark as closely as possible, the objective function is chosen such that this fits best with the objective of the benchmark. To recap, the benchmark has as objective to first minimise the number of vehicles and then to minimise the total distance. The total travel time is therefore not taken as an objective. The cost parameter for the number of vehicles is chosen such that is dominates the total distance. It was

observed that total distance after the Savings algorithm (for different objective functions) is maximally around 800 units higher than the best-known distance and therefore we choose the cost parameter for number of vehicles to be 1000. Overall, the objective function becomes as chosen in Formula (5.1).

$$\min z = 1000 \cdot \#vehicles + 1 \cdot distance + 0 \cdot time \tag{5.1}$$

The savings algorithm is run for all relevant instances of the benchmark and the results are compared with the best-known results to determine how good the results are. The results, as well as a graphical representation of some of the results, are shown in Section 5.2.1.

5.1.3 Local Search algorithm

As described in Section 4.1.6, the simulated annealing process uses a couple of parameters to control the cooling and therefore the behaviour of the algorithm. Overall, we must determine the best value for 7 parameters, namely the 4 control parameters α, k, c_{start} and c_{stop} and the 3 operator parameters, p_1 through p_3. To determine all these parameters at once is computationally not possible. We choose therefore to split the problem into different pieces which can well be solved in reasonable time.

Start and stop temperature (c_{start} and c_{stop})

The first step of setting the parameters for the local search algorithm is to determine the start and stop temperature c_{start} and c_{stop}. As described in literature, this is done using an acceptance curve. In this curve, the percentage of neighbour solutions that are accepted (acceptance percentage) is plotted against the temperature. Besides the acceptance percentage we choose to also plot the number of better solutions found. The values k and α are not important for this determination (they do not influence the result) and are set to 0.98 and 1000 respectively. Since the values for the operator parameters are also not chosen yet, they are set to 1/3 each.

We select one instance from each instance group (LC204, LRC108 and LRC207) and determine the acceptance count for temperatures between 100,000 and 0.001. We expect that for high temperatures the acceptance percentage is high (in theory ≈ 100%, but since not all neighbours are guaranteed to be feasible, that is, obey time window and capacity restrictions, this will not be the case in this experiment) and for low temperatures that the acceptance percentage is low. The start and stop temperatures are than chosen such that above the start temperature and below the stop temperature the acceptance rate is constant.

Markov Chain length (k) and temperature decrease factor (α)

The second step is to determine α and k. We want them to be such that the runtime of the algorithm is constant, and therefore the total number of iteration is constant. The number of iterations can be calculated using Formula (5.2).

$$\#iterations = k \cdot \left\lceil \log\left(\frac{c_{stop}}{c_{start}}\right) \cdot \frac{1}{\log(\alpha)} \right\rceil \tag{5.2}$$

This indicates that, given c_{stop} and c_{start} and that the number of iterations is constant, k over log(α) must be constant. To determine the influence of α on the solution result, we conducted an experiment in which we determine the number of iterations is kept constant (112,500; 450,000 and 2,250,000) by adjusting k for values of α between 0.9 and 0.99. We expect that the value of α does not influence the results, i.e. if the number of iterations is constant, the results will be comparable.

Once α is known, the only remaining control parameter is k. Since the aim of the algorithm is to find an as good as possible solution in a given amount of time, this parameter k determines the runtime

of the algorithm. The runtime however also depends on the number of order points and the parameters p_1, p_2 and p_3, because each operator has a different runtime associated with it. Since the number of order points is constant across all instances of the benchmark, we will keep this factor out of the equation. If we can associate a runtime t_1, t_2 and t_3 with the operations 1 to 3, we can write the runtime of the algorithm as a function of k and therefore k as a function of the desired runtime. This is shown in Formula (5.3).

$$runtime = \#iterations \cdot \#time\ per\ iteration \therefore$$
$$runtime = k \cdot \left\lceil \log\left(\frac{c_{stop}}{c_{start}}\right) \cdot \frac{1}{\log(\alpha)} \right\rceil \cdot \prod_i t_i \cdot p_i \quad (5.3a)$$

$$k = \frac{runtime}{\left\lceil \log\left(\frac{c_{stop}}{c_{start}}\right) \cdot \frac{1}{\log(\alpha)} \right\rceil \cdot \prod_i t_i \cdot p_i} \quad (5.3b)$$

Runtime of operators (t_1 through t_3)

To associate a runtime to the operations 1 through 3, we conducted an experiment where we varied the number of iterations between 10,000 and 1,000,000 for each operator separately, e.g. 10,000 operations with only a pointmove (p_1) for all benchmark instances, 1,000,000 operations with only an orderswap (p_3) for all benchmark instances, etcetera. From the results of these run we determined the average runtime of a single iteration, based on 24 experiments (8 instances * 3 operation counts) per measurement point.

We expect lower runtimes for instances with many vehicles, since the optimal insertion location determination is probable the bottleneck for all operations. Furthermore, we expect that a pointmove (t_1) is much faster than an ordermove (t_2) or orderswap (t_3) since there is only one trip involved when doing a pointmove. Finally, we expect that the orderswap takes at least twice as much time as the ordermove since all calculations are doubled.

Frequency of operators being used (p_1 through p_3)

To determine the values for p_1 through p_3, 1350 runs are conducted varying p_1, p_2 and p_3 between 20% and 60% and using a runtime of 10, 60 and 300 seconds. The results are compared based on difference (Δ) between measured and best known solution value, using a heat map. The Δ-values are shown both absolute and relative. We expect that a high value for p_2 will be best since the ordermove is the only operator that covers the whole solution space. The pointmove is quick and therefore probably also useful to obtain good results.

5.1.4 Modified local search algorithm

As described in Section 4.1.6, this is only the basic local search algorithm, and three more additions (modifications) are proposed to increase the quality of the results; the cold start, the introduction of a cost parameter to (temporarily) accept infeasible solutions and the Large Neighbourhood Search method. The experimental setup for each of these modifications is described here, the corresponding results are shown in Section 5.2.3.

Cold start

The local search algorithm starts with the solution found in the savings algorithm, a so called warm start (the solution is already pre-processed to be of a certain quality). As such, certain decisions about which orders are combined into trucks are already made. Although in principle using the operators the whole solution space can be searched, the warm start might already be focussed

towards a certain direction. Instead of using a warm start we can also start with a solution where every order is in a separate trip. This is similar to the start of the savings algorithm. For all the instances of the selected benchmark groups of instances, the results of the cold and warm start are compared. We would expect that the warm start performs better than the cold start especially for small runtimes, and that the results level out for high runtimes.

Since results turned out to be not as suspected, we additionally conducted an experiment with another cold start, which was created by randomly assigning orders to vehicles, to check whether the cold start with all orders in a separate truck has "special properties" which makes local search easier. It was difficult to create an additional cold start which does obeyed all restrictions, making it somewhat less obvious to compare the results.

Temporarily accepting infeasible solutions

A second modification is to temporarily allow for time window or vehicle capacity violations, by adjusting the objective function. By introducing cost parameters, the algorithm can escape local minima without having to modify the whole structure. Also, this modification is in line with many real-world applications where time window violations and even to a certain extent capacity violations are allowed at a certain financial penalty. To be able to compare the results with the benchmark, these solutions are however not used as a final result and are only used to allow quicker movement from one valid solution to another valid solution. This is accomplished by modifying the local search such that the comparison works with the solution value of the new objective function, but the solution is marked as best solution if and only if not only the objective function is lower than the current best solution, but the solution is also valid.

To reduce the computational complexity of determining two additional parameters, one for the time window violations and one for the capacity violations, we decide, strengthened by the observation that the benchmark is so designed that capacity violations are rare, to introduce only one penalty parameter c_4 which penalises both the time window violations and capacity violations. The result is shown in Formula (5.4).

$$\min z = 1000 \cdot \#vehicles + 1 \cdot distance + 0 \cdot time + c_4 \cdot (time\ window\ violation + capacity\ violation) \tag{5.4}$$

To determine the results using this new objective function, we tested this function for t=10,60 and 120 seconds for values of c_4 between 1 and 100,000. We expect to see the best results when c_4 is not too high, since for high c_4 there is no difference with the current objective function, and not too low, because then it becomes difficult for the local search to switch back to a valid solution.

Large Neighbourhood search

The third and final modification to the local search algorithm is to implement an additional operator, Large Neighbourhood Search. The principle of LNS is already discussed in Section 4.1.6, but was until now not used in the algorithm. As described, LNS comes with an additional parameter, the destruction factor df_{LNS}. The optimal value for the destruction factor will be determined in combination with the fraction p_4 this operator is used. However, before we can run experiments to determine what the best values for p_4 and df_{LNS} are, we must first determine the time associated with this operation, t_4; because this is a parameter in the formula to determine k (5.3b).

To determine the runtime of one iteration of LNS, we performed 160 experiments doing 20,000 iterations of LNS for different destruction factors (between 0.1 and 0.5), both for LRC1 and LRC2 series of the benchmark, since we will see (Section 5.2.2, Table 5.n) that there is a large difference in runtime with few and with many vehicles.

Now that we have a correlation between the destruction factor df_{LNS} and the time per operation, we can calculate for a range of p_4 values (0.1 to 0.4) and a range of df_{LNS} values (0.05 to 0.35) which parameter setting yields the best results and whether it performs better than without LNS. To keep the probabilities of each operator cumulatively equal to 1, we multiply p_1 through p_3 with $(1-p_4)$. Therefore, the ratio between p_1, p_2 and p_3 remains the same.

We expect, based on literature (Ropke, 2005), that LNS leads to a performance gain. Furthermore, probably he optimal p_4 and df_{LNS} will be low since LNS is quite time consuming, especially for high destruction factors. Since the results of the LNS where not as expected (see Section 5.2.3) and this is probably caused by the high runtime of a single iteration of LNS, we performed an additional experiment keeping the number of iterations constant to eliminate the influence of the runtime (and therefore the length of the Markov chain) on the results. In this experiment, we used the number of iterations equal to that of 60 seconds' calculation without LNS and used a p_4 of 0.3 and a df_{LNS} of 0.35.

5.1.5 Vehicle routing problem for 'Upload!' participants

Sections 5.1.3 and 5.1.4 described the parameterisation of the local search algorithm given the objective of the benchmark, to foremost minimise the number of vehicles and as second objective to minimise the distance. These objectives are not necessarily the objectives of transportation companies targeted for participation in 'Upload!'. These companies have a fixed fleet, and therefore require solutions fitting the number of vehicles available. Furthermore, are the utility costs for trucks on average about equal to the costs of requiring a driver at the truck. Time window violations are sometimes allowed, given a penalty per minute of time window violation.

In this section, we describe which experiments we conduct to show how the proposed algorithm performs for these situations. For simplicity, and fairness of comparison with the benchmark, time windows remain hard, i.e. in the proposed solution no time window violation may occur. Besides that, the cost parameters are adjusted to better compare with real life problems.

Since the exact values of the cost parameters do not matter (since distances are no real kilometres, the values do not relate to a certain scale) we choose the following objective function:

$$\min z = 10 \cdot \#vehicles + 0.5 \cdot distance + 0.5 \cdot time + 100 \cdot (time\ window\ violation + capacity\ violation) \quad (5.5)$$

with the additional constraint that the final result may not contain any time window nor capacity violation.

Because the objective function changes, it is to be expected that the acceptance curve might also change. To determine the new values for c_{start} and c_{stop} we conduct an experiment like that described in Section 5.1.3 – heading start and stop temperatures. Again, we choose the instances LC204, LRC108 and LRC205 as representatives for the benchmark group.

We assume that changing the objective function does not have an impact on the optimal value of α nor on the time required to perform one iteration of each of the operators (t_1 through t_3). As such the last remaining step of the parameterization is to determine the best values for p_1 trough p_3. We compared the results with the best-known results given the objective function in (5.5) for all relevant instances of the benchmark at t=10,60 and 120 seconds. Since this is a different objective function than the benchmark is designed for, it is now possible that better solutions are found using this algorithm than the best solution to the benchmark.

We expect that relative difference between our results and the one of the benchmark will decrease, since the “best known results” are now no longer certainly the “best known results” because of the different objective function.

5.1.6 Selection of inefficient trips and orders

The second step of the ‘Upload!’ algorithm is to determine which trips and orders cause inefficiency. These trips are than selected for exchange on a marketplace. In this section, we describe the method used to determine which trips and orders cause inefficiency.

Selection of inefficient trips

As described in Section 4.2.2, we first determine which trips cause inefficiency and from these trips the causing orders are selected. In this way, the total computation effort is decreased. To determine the efficiency of trips, we determine the financial and empirical performance indicators (PIs) as described in Section 4.2.2. To recap, the financial PIs are calculated using the ratio between the cost of executing the trips (which follows from the objective function) and the gain of executing the trip, which is calculated as the distance of the trip multiplied by the load transported over that distance, in other words, we use a fixed financial gain per unit of distance per unit of volume transported (Formula 4.3c).

The empirical PIs of the trip are based on the load factor (that is the percentage of volume of the vehicle that is used on each leg of the trip), the waiting time during the trip (due to the time window constraints), the number of orders in the vehicle and the distance gain due to combining trips (which is parallel to the objective of the savings algorithm; the distance required to deliver every package in a separate truck divided by the distance traversed by the truck, e.g. the degree of coinciding between orders of the same trip).

To determine the characteristics of inefficient trips, we determine if there is a correlation between the empirical and financial PIs and whether the trips with bad performance relate to solutions with high objective values compared to the best-known solution values. To do so, we analysed both the financial and empirical PIs for all solutions found in Section 5.1.5, for all values of p_1 to p_3 and for all times, including the results of the Savings Algorithm. In total, this means that we examine the PIs for 7,258 trips, with various degrees of expected performance. Based on these results, we determine which trips are marked inefficient.

Selection of inefficient orders

To determine which orders cause inefficiency, we planned to remove sets of up to 4 orders from the trips and determine the increase in performance of the new trip bearing these order points. The required computation time to do so however is very large. Take for example a trip of 40 orders, which is about the upper bound of orders per trip found in this research, to enumerate all combinations of orders with up to 4 orders this equals 2,254,240 possible combinations (see Formula 5.6) for calculation.

$$\#\ of\ iterations = \sum_{n=1}^{N} \frac{M!}{(M-n)!} \tag{5.6}$$

where M is number of orders in trip and N is the number of combinations to remove.

For one inefficient trip we perform this full iteration to get some insight in what is likely to happen. We expect that most of the inefficient trips will consist of only a few orders, and in that case this problem will not occur. For the trips with many orders we expect that only the first of last orders of

the trip can be removed, because removing orders from the middle of the trip will only introduce more waiting time at the end the trip.

5.1.7 Vickrey-Clarke-Groves auction

To determine how well the Vickrey-Clarke-Groves auction (VCG auction) performs, we conducted two experiments exchanging the inefficient orders selected using the selection criteria for inefficient orders.

The first experiment is conducted to determine the runtime of the algorithm depending on the number of companies participating in the auction. To determine the runtime, we grouped together at random increasing amounts of solutions containing at least one inefficient trip. We calculated the average number of inefficient orders from the total of these companies and then let the algorithm run. The time to calculate the bid-price b_{ij}, the best allocation determination and the actual price is than calculated for each of the instances. We expect an exponentially increasing runtime on all three runtime calculations. Furthermore, we expect that the maximum number of participants of the auction will be rather small, because runtimes will quickly become too big to perform.

The second experiment focuses on the locations of the depots of the auction participants. Since all trips of all participants must start and end at the depot, we expect that if the overlap of the trips conducted by the companies is high, the possibilities to exchange will be high, whereas if there is no overlap in trips, exchange is difficult. The measure of overlap that we use is the distance between the locations of the depots, since the benchmark is designed to have orders roughly in a circle around the depot. In this experiment we use groups of four auction participants, where every participant again is one solution containing at least one inefficient trip. We arrange the depots of these participants in a circle around a centre point of the imaginary map, arranged in the four winds (NW, NE, SE and SW) and increase the distance from the centre point. Here we expect that the smaller the distance from the depots to the centre point of the map, i.e. the larger the overlap of order points, the higher the exchange potential and therefore the better the matching algorithm performs.

5.2 Results

In this section the results to the experiments described in Section 5.1 are shown and explained. The results of the savings algorithm, the local search and modified local search, the local search for 'Upload!' participants, the results of the selection of inefficient trips and orders algorithm and for the Vickrey-Clarke-Groves auction are all shown here.

5.2.1 Savings algorithm

In Table 5.2 the results of the savings algorithm are shown. The average difference (Δ) between the results and benchmark, both absolute and relative summarised at instance group level.

Table 5.2 Results savings algorithm

Instance	ΔVeh			ΔDist			Runtime (ms)
	min	max	avg	min	max	avg	
LC2	+0 (0%)	+2 (67%)	+0.9 (28%)	+0 (0%)	+215 (36%)	+89 (15%)	0.12
LRC1	+4 (36%)	+8 (62%)	+5.6 (49%)	+234 (16%)	+669 (58%)	+503 (38%)	0.05
LRC2	+1(33%)	+4 (133%)	+2.5 (87%)	+181 (13%)	+652 (61%)	+378 (35%)	0.14

The results for two typical instances (one where the savings algorithm gives a good result and one where there is still a lot of room for improvement) are shown in Figure 5.1. In grey the best edges are shown and in colour the different trips are indicated. For instance LC205 the best-known number of

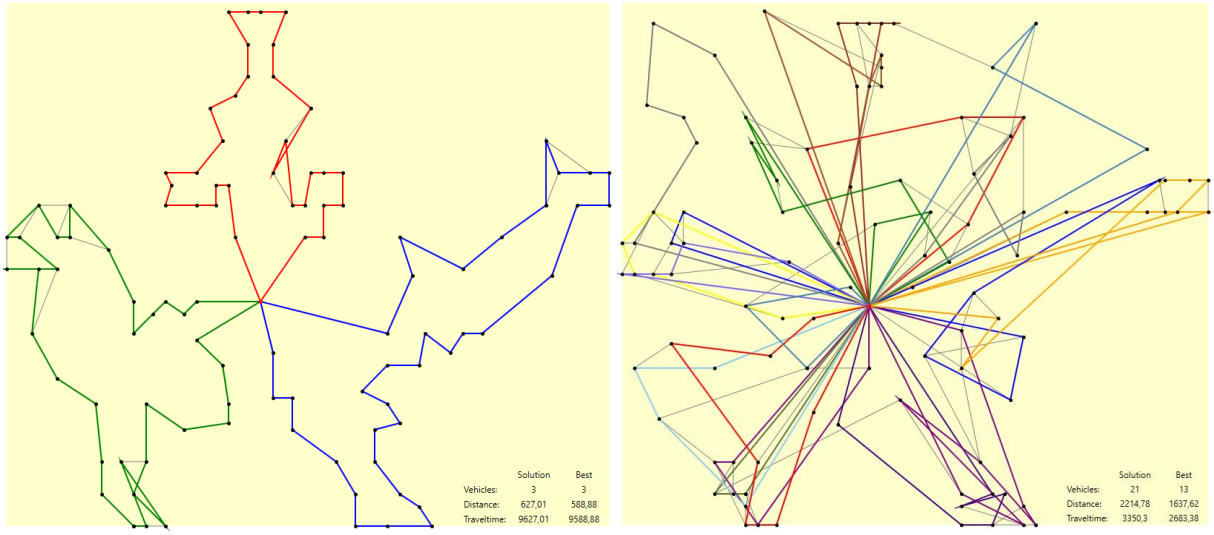


Figure 5.1 Visualisation of savings algorithm result for a) instance LC205 and b) instance LRC105

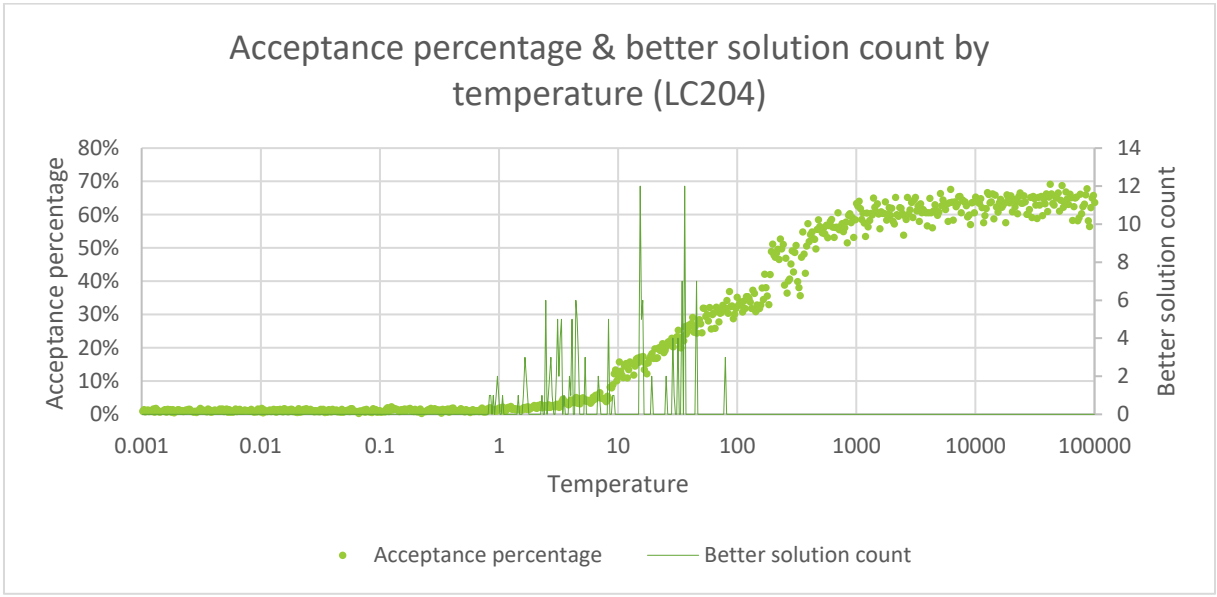
trips is 3, which is identical to the result found by the savings algorithm. The chosen routing is also very similar to that of the best-known solution. For instance LRC105 on the other hand, the savings algorithm needs 21 routes whereas the best-known solution only requires 13 routes. Here the solution of the savings algorithm is not sufficient and a second algorithm is required to improve the performance.

5.2.2 Local Search algorithm

In this section the results of the local search parameterisation are shown. The headers follow the same pattern as the experiment description in Section 5.1.3.

Start and stop temperature (c_{start} and c_{stop})

The results of the acceptance count experiments are shown in Figure 5.2.



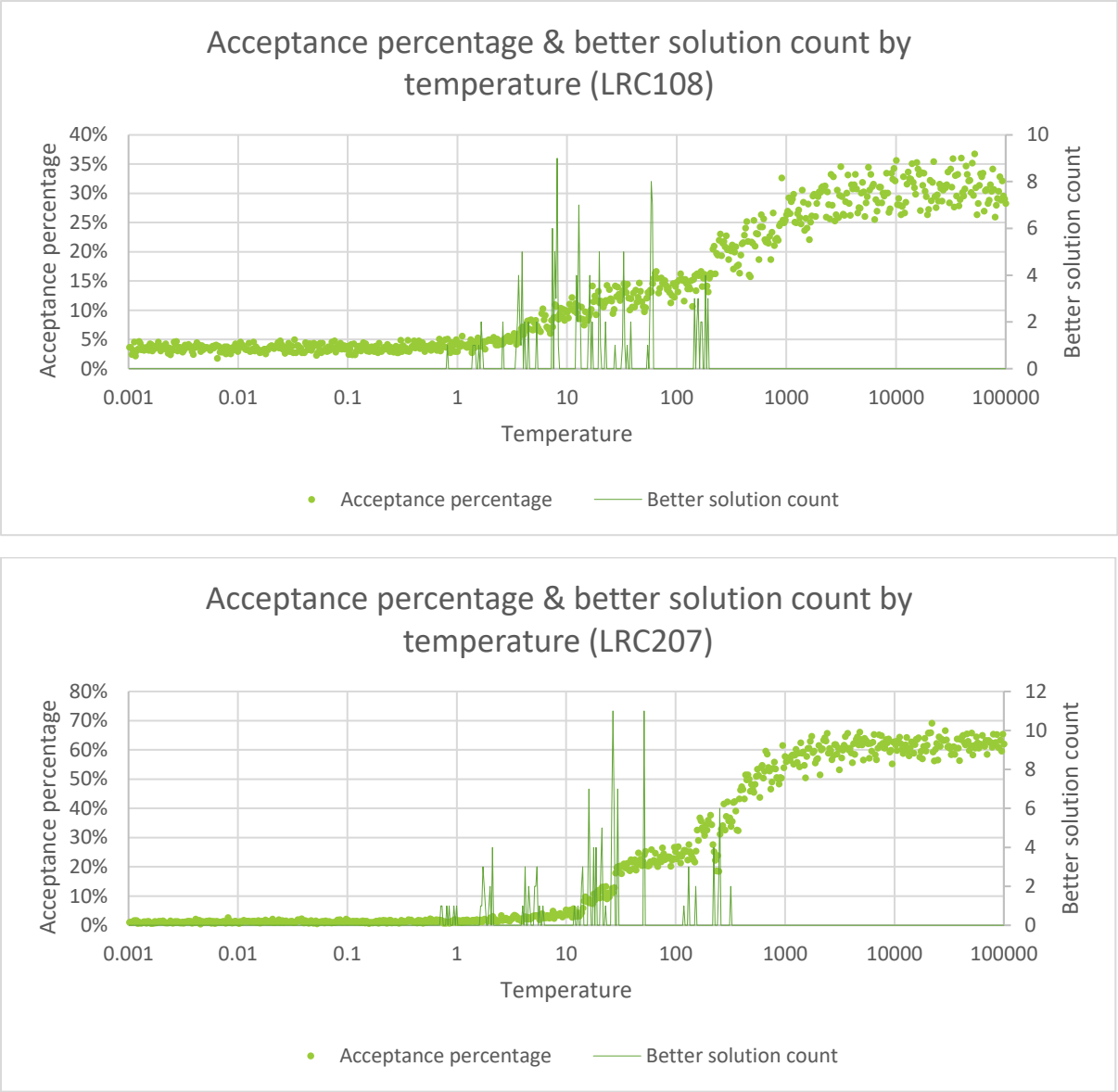


Figure 5.2 Acceptance percentage and better solution count to determine c_{start} and c_{stop} for 3 instances.

In these figures we see that, as expected, for high temperature the acceptance percentage is high and for low temperatures the acceptance percentage is low. We also see that the curves are smooth, which indicates that the objective function is smooth. We also see that the acceptance count never reaches 100% because not all neighbours can be accepted. If the operation leads to an infeasible solution, either because of time window violation or because of capacity violation (the operators are defined such that precedence relation violation is not possible), the solution is not accepted, independent of the temperature. Based on these results we choose that c_{start} = 10,000 and c_{stop} is 0.1 because at temperatures above 10,000 no real changes happen to the curve and the same holds for temperatures below 0.1.

Markov Chain length (k) and temperature decrease factor (α)

The results of the experiment of the influence of α on the solution values are shown in Figure 5.3.

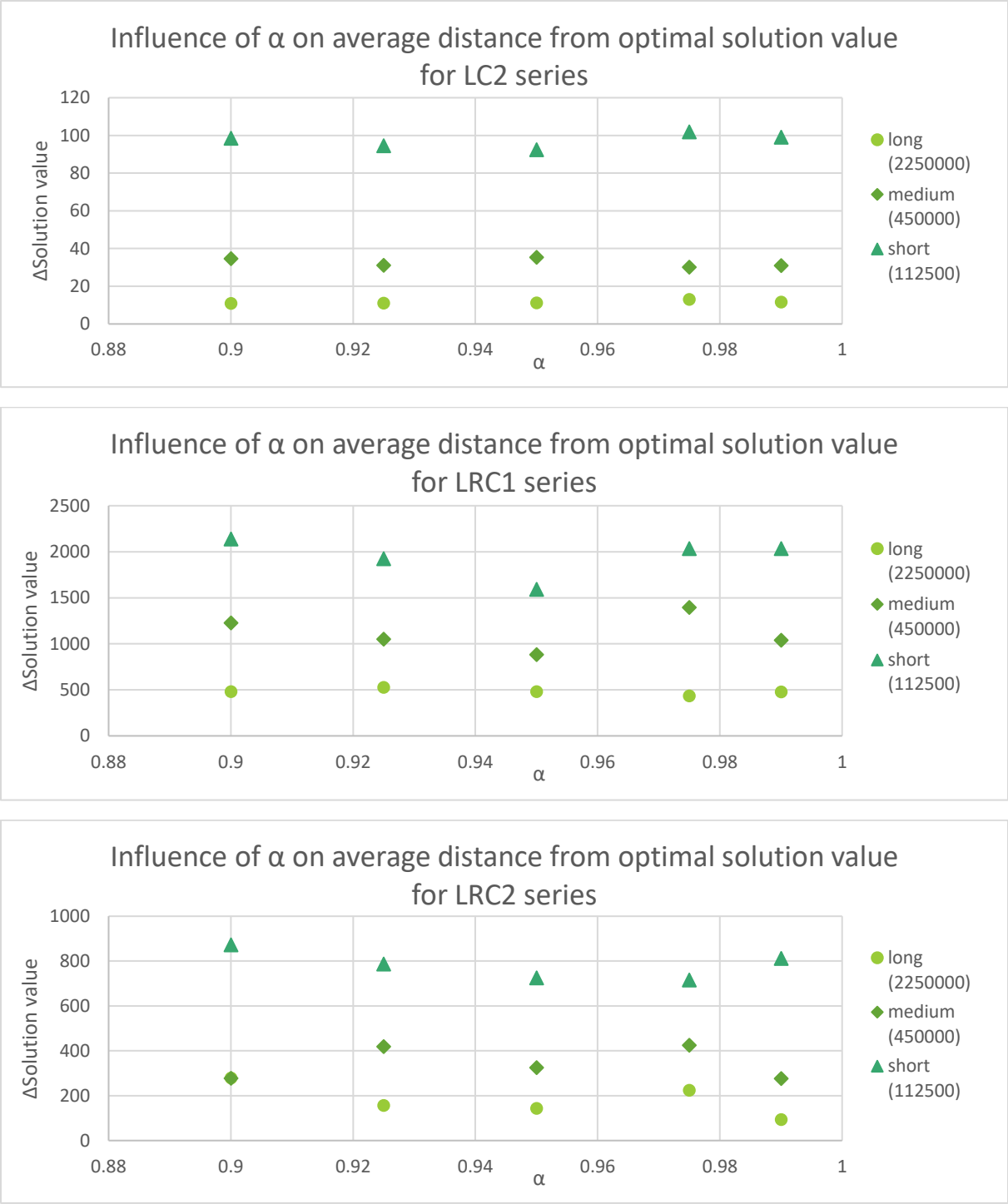


Figure 5.3 Influence of α on distance from optimal solution value for different benchmark series

In these figures we see that, as expected, the difference between the best-known solution value and the obtained solution value (Δ solution value) decreases if the algorithm can do more iterations. However, the exact value of α does not seem to make much difference. If the total number of iterations is kept the same, the results remain more or less the same. For the rest of the calculation we choose α to be equal to 0.95.

Runtime of operators (t_1 through t_3)

The results of the determination of the runtime of operators are shown in Table 5.3.

Table 5.3 Iteration time determination

Instance	Order points	Order point layout	Vehicle capacity	t_1 (ms)		t_2 (ms)		t_3 (ms)	
				AVG	STDEV	AVG	STDEV	AVG	STDEV
LC2	100	Clustered	High	0.011	<0.001	0.077	0.015	0.269	0.042
LRC1	100	Mixed	Low	0.001	<0.001	0.020	0.003	0.023	0.006
LRC2	100	Mixed	High	0.013	0.001	0.081	0.021	0.273	0.107

The values t_1 through t_3 appear to be independent of the order point layout, as the results for LC2 and LRC2 are comparable but do largely depend on the vehicle capacity; and therefore on the number of trucks used. We also see that the ordermove is about 7 times as slow as the pointmove and that the orderswap is about 3 times as slow as the ordermove. The first is as expected, the second is slightly off expectation, since the orderswap is computational comparable with the ordermove. Apparently the orderswap introduces some overhead in this implementation. For further research, this indicates that there is runtime optimization potential.

For the remainder of the thesis, we use the values in the columns AVG as operation runtime for the respective operators and instance groups.

Frequency of operators being used (p_1 through p_3)

The results of the determination of p_1 to p_3 are summarized in a heat map (in this case, green values correspond with good values and red values correspond with bad values). In this heat map the values of p_1 are shown vertically and the values of p_2 horizontally. The value of p_3 is a result of these other two percentages since it must add up to 1. The different instances are shown vertically, repeating p_1; the time is shown horizontally repeating p_2, see Figure 5.4.

Δ Solution value

		t = 300 s					t = 60 s					t = 10 s				
	p2 \ p1	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6
LC2	0,2	41,8	30,6	17,9	17,9	33,7	29,8	31,3	26,4	26,4	26,2	214,6	315,7	125,4	40,9	320,2
	0,3	29,7	18,2	17,9	**17,9**		30,9	25,6	29,8	**22,9**		303,6	210,1	201,2	**42,1**	
	0,4	25,9	25,8	25,8			30,3	29,8	17,3			301,9	124,0	202,0		
	0,5	26,0	17,9				21,9	21,9				35,0	203,6			
	0,6	25,8					29,8					122,0				
LRC1	0,2	439	439	348	438	526	706	619	873	710	537	1033	2075	1067	1376	1033
	0,3	523	438	535	**525**		702	972	1073	**625**		1377	1720	1379	**1034**	
	0,4	524	351	437			878	713	782			1032	1046	1380		
	0,5	436	523				795	700				1032	1381			
	0,6	609					620					1382				
LRC2	0,2	368	350	237	353	346	555	537	368	270	380	1336	988	1056	872	804
	0,3	355	349	361	**244**		475	272	461	**270**		1331	857	962	**603**	
	0,4	235	345	230			465	276	298			1112	1276	834		
	0,5	346	348				370	311				976	593			
	0,6	356					372					1125				

Relative Δ Solution value

		t = 300 s					t = 60 s					t = 10 s				
	p2 \ p1	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6
LC2	0,2	1,2%	0,9%	0,5%	0,5%	0,9%	0,8%	0,9%	0,7%	0,7%	0,7%	6,0%	8,8%	3,5%	1,1%	8,9%
	0,3	0,8%	0,5%	0,5%	**0,5%**		0,9%	0,7%	0,8%	**0,6%**		8,5%	5,9%	5,6%	**1,2%**	
	0,4	0,7%	0,7%	0,7%			0,8%	0,8%	0,5%			8,4%	3,5%	5,6%		
	0,5	0,7%	0,5%				0,6%	0,6%				1,0%	5,7%			
	0,6	0,7%					0,8%					3,4%				
LRC1	0,2	3,4%	3,4%	2,7%	3,4%	4,1%	5,5%	4,8%	6,8%	5,5%	4,2%	8,0%	16%	8,3%	11%	8,0%
	0,3	4,1%	3,4%	4,2%	**4,1%**		5,5%	7,5%	8,3%	**4,8%**		11%	13%	11%	**8,0%**	
	0,4	4,1%	2,7%	3,4%			6,8%	5,5%	6,1%			8,0%	8,1%	11%		
	0,5	3,4%	4,1%				6,2%	5,4%				8,0%	11%			
	0,6	4,7%					4,8%					11%				
LRC2	0,2	8,4%	8,0%	5,4%	8,1%	7,9%	13%	12%	8,4%	6,2%	8,7%	30%	23%	24%	20%	18%
	0,3	8,1%	8,0%	8,2%	**5,6%**		11%	6,2%	11%	**6,2%**		30%	20%	22%	**14%**	
	0,4	5,4%	7,9%	5,3%			11%	6,3%	6,8%			25%	29%	19%		
	0,5	7,9%	7,9%				8,5%	7,1%				22%	14%			
	0,6	8,1%					8,5%					26%				

Figure 5.4 Heat map (absolute and relative) of Δ solution values for different values of p_1 and p_2 and for different runtimes and benchmark groups.

In this figure we see that, again, a longer algorithm running time gives better results. Furthermore, we see that the LC instance is easier to solve for this algorithm than the LRC instances; the relative difference in solution value is much higher in case of the LRC instances. With respect to the best values for p_1 through p_3, we are looking for a combination that gives structurally good results. By averaging all different combinations, we see that values of 30%, 50% and 20% for respectively p_1, p_2 and p_3 give the best results. This is in accordance with our expectations. Remember that the second operation, the ordermove, is the only operation that is required to cover the whole solution space. The pointmove operator (associated with p_1) has the advantage that it is very quick and therefore an

easy way to improve. The values for p_3 (which are constant when looking over the diagonals of the heat map) show that although the best result is found when p_3 is the lowest value tested, lower values for p_3 are not likely to give better results, since there is no clear trend that all values become better when moving to the left upper corner.

Overall, we see that we can find solution values within about 5% from the optimal value after 5 minutes of runtime for the most difficult case of mixed order points configuration and many trips per

5.2.3 Modified local search algorithm

In this section the results of the modified local search algorithms are shown. The headers follow the same pattern as the experiment description in Section 5.1.4.

Cold start

In Table 5.4 the comparison between a cold and warm start is shown), where Δz is the difference between the average with a warm start and the average with a cold start (a negative number indicates that the cold start performed better).

Table 5.4 Comparison between cold and warm start

Instance	Order points	Order point layout	Vehicle capacity	t = 1s	t=5s	t = 10s	t=60s	t=120s	t=300s
				Δz	Δz	Δz	Δz	Δz	Δz
LC2	100	Clustered	High	640	-56	-150	1	0	0
LRC1	100	Mixed	Low	8762	20	-152	135	55	12
LRC2	100	Mixed	High	2132	-32	-64	132	80	22

This table shows some remarkable results. The results for runtimes of 1 second and 60+ seconds are all as expected. The warm start performs better, and the influence of the start decreases as the runtime increases. The negative results indicate that for runtimes of 5 and 10 seconds the cold start performs exceptionally good, even outperforming the warm start. Apparently, the cold start gives the solution method an edge to find better results than the warm start does. One possible explanation is that 5 or 10 seconds gives the algorithm too little possibilities to escape from the local optimum found with savings algorithm.

In Table 5.5 we added the results of the alternative cold start, as described in Section 5.1.4.

Table 5.5 Comparison between alternative cold and warm start

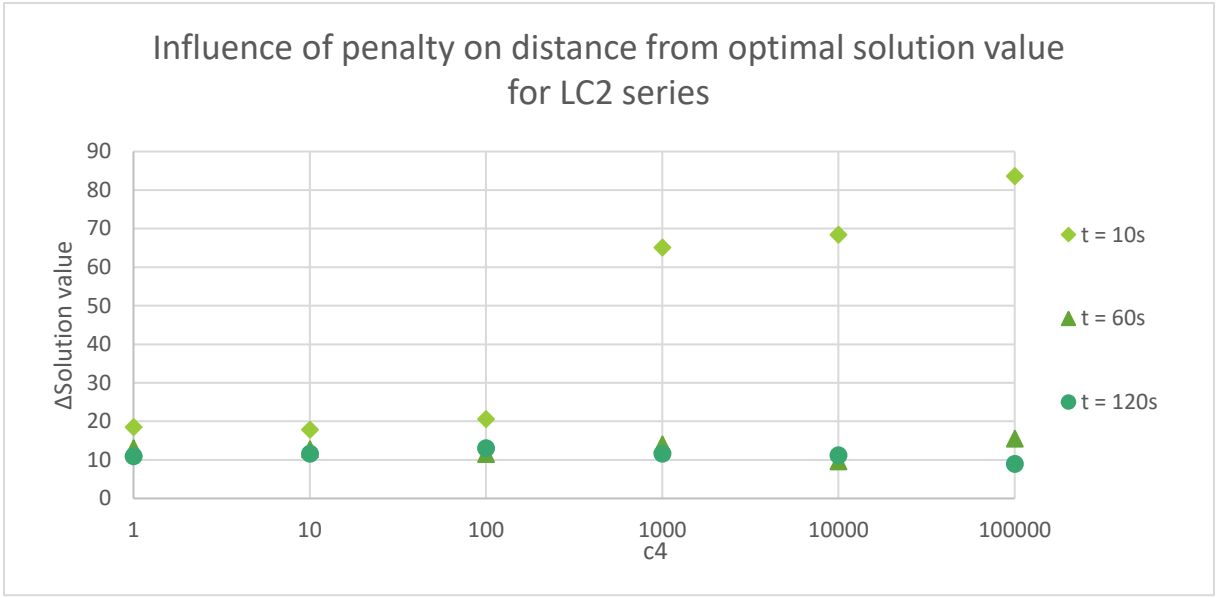
Instance	Order points	Order point layout	Vehicle capacity	t = 1s	t=5s	t = 10s	t=60s	t=120s	t=300s
				Δz	Δz	Δz	Δz	Δz	Δz
LC2	100	Clustered	High	1265	1056	520	42	18	0
LRC1	100	Mixed	Low		6542	1258	632	122	34
LRC2	100	Mixed	High		3621	1122	583	165	55

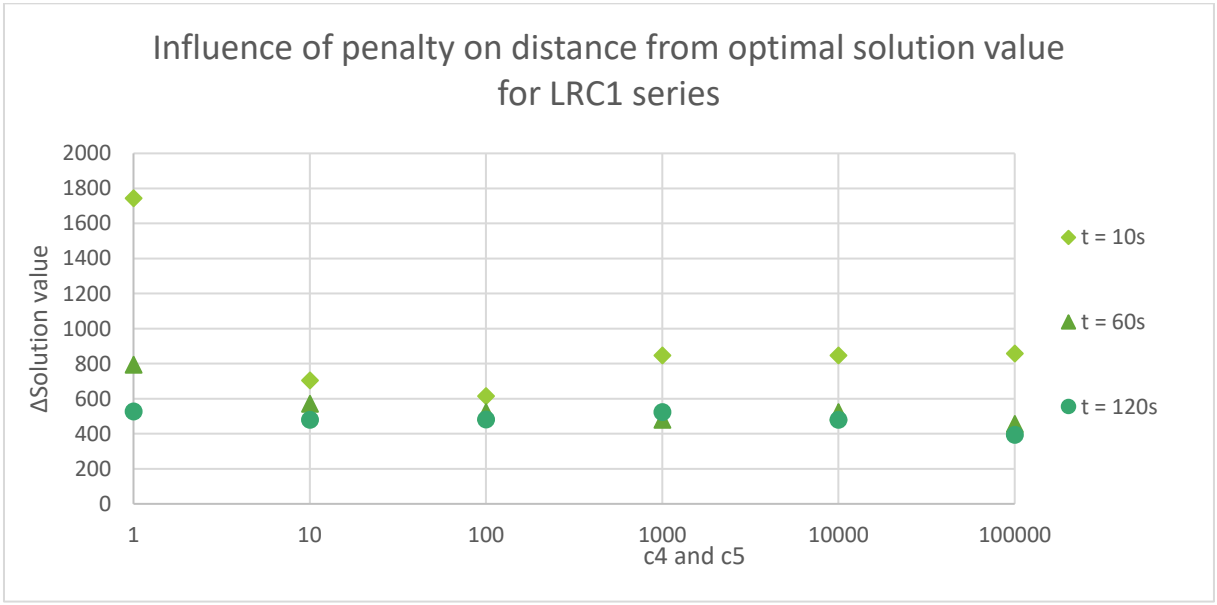
The results shown in this Table are compatible with our expectations of a cold start, the cold start performs always worse than the warm start. After 1 second, the algorithm did not found a feasible solution for all instances of the LRC1 and LRC2 benchmark group, and therefore these measurements are not shown in the table. This result proves that there is a special characteristic about the cold start when all trips consist of one order.

Overall, we conclude that for most real-life applications the warm start will perform better or at least as good as the cold start and the warm start will therefore be used in the rest of the calculations.

Temporarily accepting infeasible solutions

The modified objective function was tested for different runtimes and the difference in solution values between the best-known value and the obtained value where plotted (Figure 5.5) for different values of the penalty parameter c_4. Note that the value 100,000 corresponds with the situation before introducing this extra cost parameter, since the situation without a penalty was modelled as 100,000 expecting that such a high penalty would mean that no violations would ever be used as intermediate results.





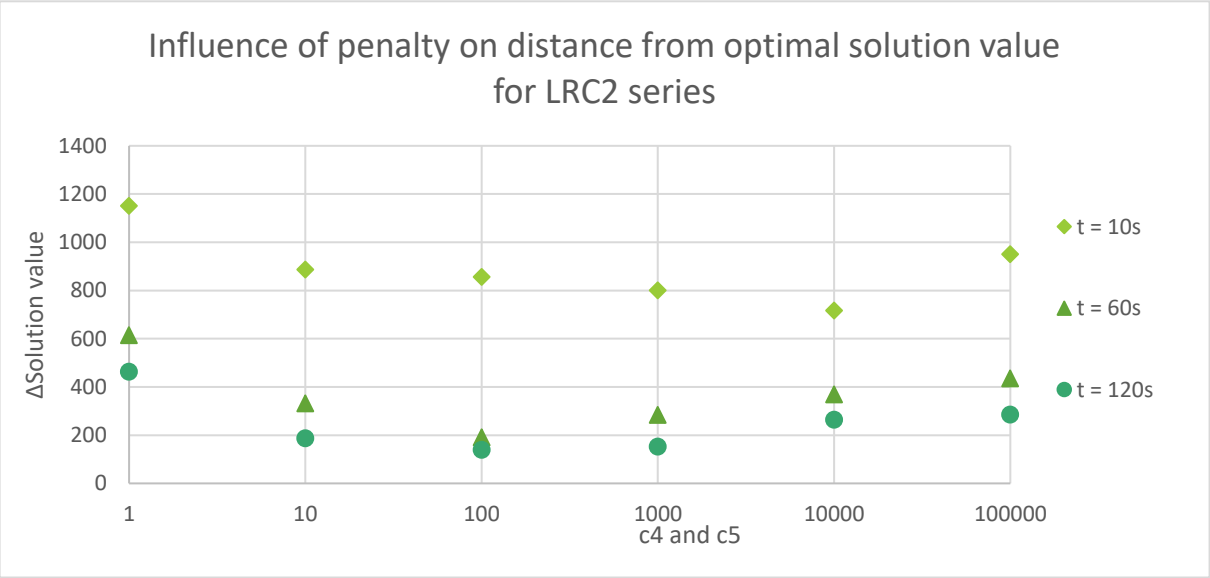


Figure 5.5 Influence of cost penalty on distance from optimal solution value for different benchmark series

From this figure we conclude that introducing a cost factor does have a possible influence on the performance of the algorithm. When the cost parameter is high this corresponds with the situation when there is no possibility to use infeasible solution as intermediate results. This gives sub-optimal results. The same can be said if the cost factor is too low, presumably because there is not enough incentive for the algorithm to construct a feasible solution.

Furthermore, we also see that in some situations the solutions with runtime 60 seconds are better than the solutions after 120 seconds of runtime. These exceptions are caused if one of the 8 instances in the benchmark series happens to need an additional vehicle, which causes the average solution value to increase by 125 points. The distance is lower for all instances after 120 seconds, and as such, this phenomenon is caused by random chance.

Another interesting feature is that for the LC2 series we see a big jump in Δ solution value when moving from c_4 = 100 to c_4 = 1000. For multiple instances of the LC2 series, a considerable distance improvement is made when c_4 is lower than 100. This improvement is unique to the LC2 series and unique to the smallest time set. Apparently, the algorithm can find a better solution after a short amount of time if it is allowed to make a certain move that cannot be made if the penalty is too high. At first sight it might seem that there is a relation between the c_4 being equal to 1000 means that c_4 = c_1, but since the number of vehicles found in each solution (time and penalty parameter) is equal, this does not seem compatible. At least this result proves that the introduction of the cost penalty can make a difference.

The best results are obtained when the penalty cost parameter c_4 is equal to 100, which performs best in every instance series of the benchmark and will therefore be used in the remainder of this thesis.

Large Neighbourhood search

To determine the quality of LNS, we first determined the duration of one iteration of LNS as a function of the destruction factor df_{LNS}. The results and linear regression are shown in Figure 5.6.

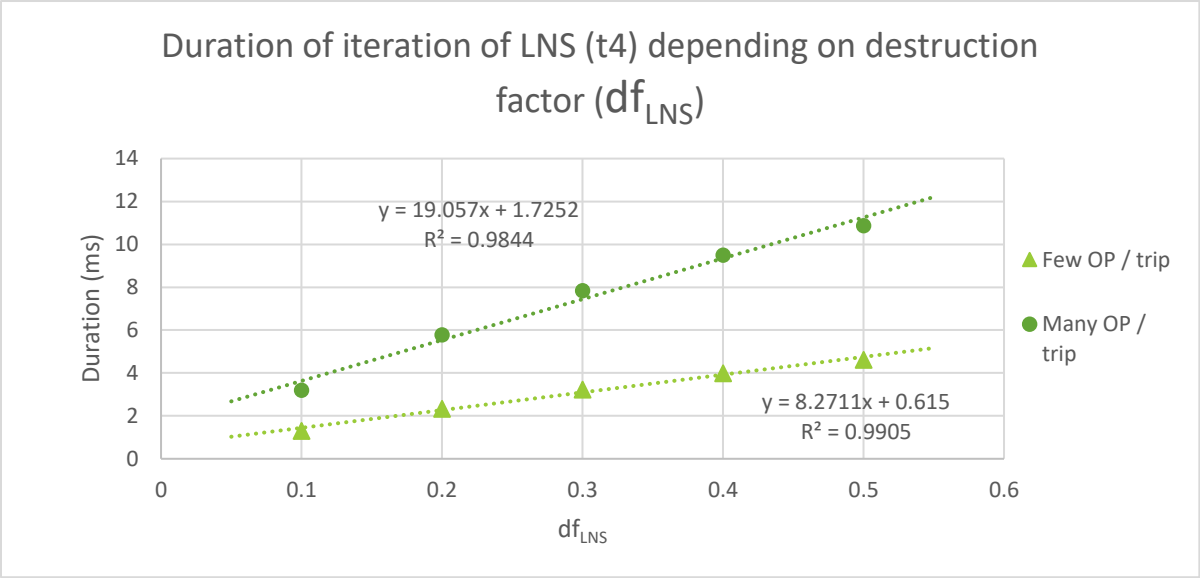


Figure 5.6 Correlation between runtime and destruction factor for few and many order points per trip

We see that again, like the determination of t_1 through t_3, the more order points there are per trip, the longer the runtime of one iteration. Linear regression shows a clear (linear) correlation between the destruction factor and the algorithm, which is as expected given that a higher destruction factor means more time-consuming rebuilding of the solution. We also see that the runtimes of LNS iterations are much higher than that of the other operators, which might indicate that LNS will not perform so good since it reduces the total amount of iterations considerable. For the Markov chain length determination with LNS we use the equations found using linear regression for few and many order points per trip as function, i.e. for an LRC2 series, we use $t_4 = 19.057 \cdot df_{LNS} + 1.7252$.

In Figure 5.7 we see a heat map (again where green values indicate good scores and red values indicate bad scores) showing the relative difference in solution value between the found value and the best-known value for the different p_4 and destruction values.

Relative Δ solution value

		t=120 s				t=60 s				t=10 s			
	p_4 / df_{LNS}	0,1	0,2	0,3	0,4	0,1	0,2	0,3	0,4	0,1	0,2	0,3	0,4
LC2	0,05	0,0%	0,1%	0,0%	0,0%	0,1%	0,0%	0,0%	0,0%	3,1%	1,5%	1,6%	4,4%
	0,15	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	1,4%	0,0%	1,7%	4,3%	5,7%	2,9%
	0,25	0,0%	0,0%	0,0%	0,0%	1,4%	1,4%	0,0%	1,5%	3,0%	1,5%	1,5%	4,4%
	0,35	0,1%	0,0%	0,1%	0,0%	0,0%	0,0%	1,5%	0,0%	3,0%	1,5%	3,0%	1,6%
LRC1	0,05	3,4%	4,4%	5,3%	5,7%	4,4%	5,0%	6,0%	6,4%	8,1%	11%	8,9%	11%
	0,15	3,7%	4,0%	4,0%	5,4%	4,7%	4,3%	4,7%	5,0%	7,8%	6,1%	8,2%	8,1%
	0,25	4,0%	4,7%	4,0%	4,0%	4,7%	4,1%	6,4%	5,0%	8,5%	8,5%	8,9%	8,5%
	0,35	4,4%	4,7%	4,7%	4,4%	5,4%	4,7%	5,7%	5,1%	9,2%	11%	9,4%	10%
LRC2	0,05	4,2%	6,1%	7,3%	9,5%	7,3%	11%	11%	12%	24%	25%	19%	27%
	0,15	7,1%	8,1%	4,0%	8,1%	8,3%	7,0%	10%	9,2%	18%	20%	22%	16%
	0,25	6,0%	8,0%	10%	7,2%	7,0%	9,3%	6,0%	12%	23%	17%	17%	19%
	0,35	4,1%	4,9%	9,0%	9,0%	8,0%	7,3%	7,2%	10%	19%	16%	17%	16%

Figure 5.7 Δ solution values for different values of p_4 and df_{LNS} and for different runtimes and benchmark series.

From Figure 5.7 we see that there is no clear trend in the results of the LNS run. At higher runtimes, it seems that a small p_4, combined with a low destruction factor performs best. For small runtimes however, this is not valid and for the LRC2 series of instances at t=10 high destruction factors seem favourable. We do see that the best values found using LNS are better than the values found without LNS and without introducing the cost parameter p_4 (compare Figure 5.4). To show the influence of LNS we also created a heat map comparing the absolute values found using LNS with the absolute value found using the optimal cost parameter $c_4 = 100$. The results are shown in Figure 5.8.

Absolute percentpoint improvement from c_4 to LNS

		t=120 s				t=60 s				t=10 s			
	p_4 / df_{LNS}	0,1	0,2	0,3	0,4	0,1	0,2	0,3	0,4	0,1	0,2	0,3	0,4
LC2	0,05	0,36	0,30	0,36	0,36	0,27	0,32	0,32	0,32	-2,48	-0,95	-1,07	-3,87
	0,15	0,36	0,36	0,36	0,36	0,32	0,32	-1,11	0,32	-1,17	-3,70	-5,09	-2,32
	0,25	0,36	0,36	0,36	0,34	-1,12	-1,09	0,31	-1,14	-2,38	-0,91	-0,97	-3,83
	0,35	0,30	0,36	0,29	0,36	0,32	0,31	-1,15	0,32	-2,39	-0,92	-2,43	-1,07
LRC1	0,05	0,31	-0,65	-1,61	-1,99	-0,32	-0,96	-1,96	-2,31	-3,38	-5,74	-4,12	-6,63
	0,15	0,05	-0,30	-0,31	-1,63	-0,62	-0,28	-0,62	-0,94	-3,04	-1,35	-3,40	-3,36
	0,25	-0,28	-0,98	-0,29	-0,30	-0,65	0,00	-2,30	-0,99	-3,78	-3,75	-4,09	-3,76
	0,35	-0,65	-0,97	-0,97	-0,64	-1,31	-0,68	-1,63	-1,01	-4,42	-5,82	-4,60	-5,56
LRC2	0,05	-1,00	-2,87	-4,09	-6,35	-2,96	-6,33	-6,89	-7,16	-4,92	-5,85	0,32	-7,13
	0,15	-3,92	-4,91	-0,80	-4,92	-3,96	-2,60	-5,66	-4,80	1,73	-0,73	-2,30	3,13
	0,25	-2,85	-4,77	-7,02	-3,97	-2,69	-4,98	-1,67	-7,82	-3,08	2,95	2,06	0,71
	0,35	-0,93	-1,76	-5,82	-5,85	-3,69	-2,95	-2,85	-5,81	0,21	4,03	2,09	3,77

Figure 5.8 Percentage point improvement due to introduction of LNS compared to introduction penalty factor

When we compare the results with the results found in the run with the cost penalties, we see that LNS performs better only in specific situations, most notably for the LRC2 instances with low runtime, and the LC2 at high runtime. The results seem counterintuitive given the good results of LNS (Ropke, 2005) found in literature. The bad results obtained are probably due to the negative influence of long runtimes required for one iteration of LNS. For a destruction factor of 0.3 e.g., the duration of one LNS operation is between 3 and 8 ms., whereas one orderswap operation takes between 0.02 and 0.08 ms. In other words, 100 orderswap operations can be executed in the same time as one LNS operation. This long runtimes, probably due to the method of implementation, is most likely responsible for the bad performance of LNS. To test this, we conducted an additional experiment for two LRC1 instances keeping the number of iterations constant. The results are shown in Table 5.6.

Table 5.6 Results for timed and untimed LNS compared to situation without LNS.

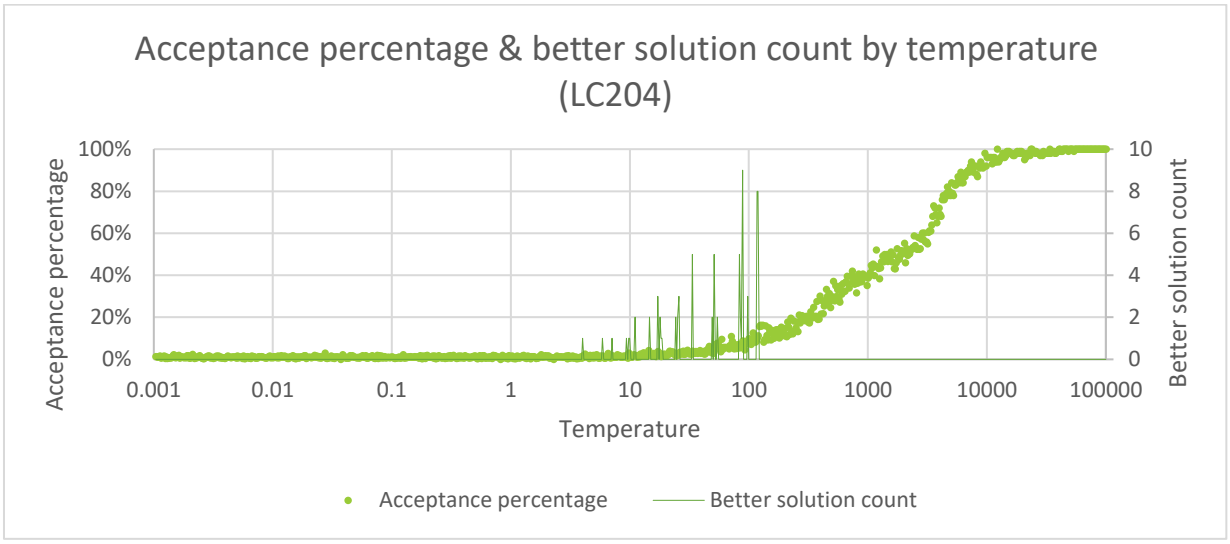
	without LNS			**Par LNS**		**timed LNS**			**untimed LNS**		
	Δ Best	k	Runtime	p_4	df_{LNS}	Δ Best	k	Runtime	Δ Best	k	Runtime
LC2	0,32%	2915	64,0	0,3	0,35	1,47%	103	66,9	0,00%	2915	1795
LRC1	4,05%	2517	52,5	0,3	0,35	5,68%	250	72,0	2,51%	2517	725
LRC2	4,36%	2915	62,7	0,3	0,35	7,21%	103	62,8	3,11%	2915	1777

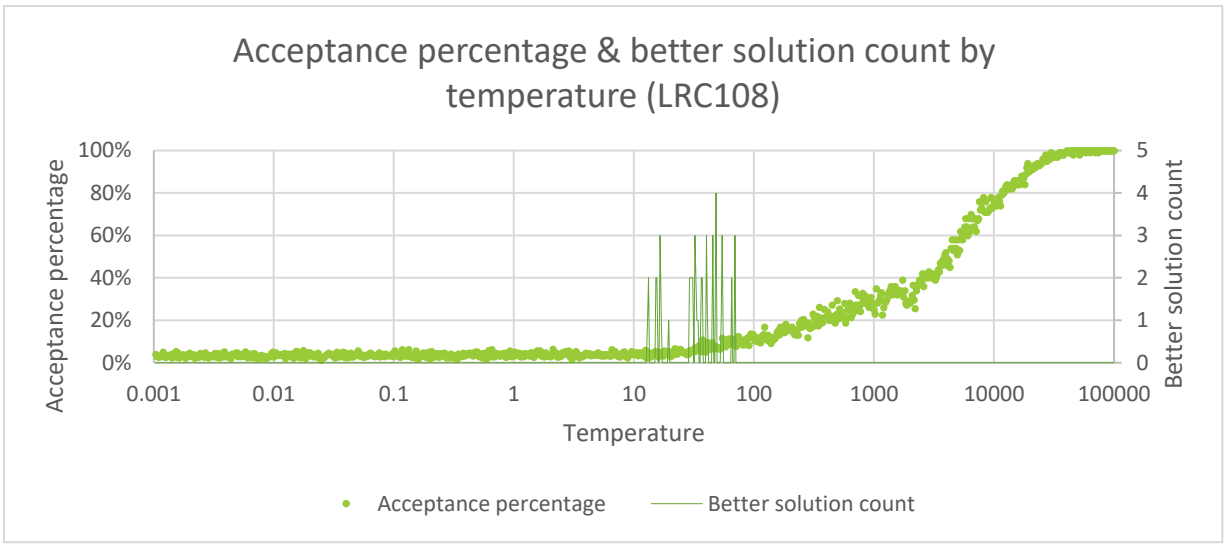
From this table we conclude that the instances with LNS are indeed much slower, but are able to produce better results when given the same amount of iterations as the regular local search. This is

in line with our expectations of the LNS algorithm. Because runtime is an important criterion, and there was no time to further improve the implementation of the LNS algorithm, in the remainder of this thesis we will not use the LNS operator.

5.2.4 Vehicle routing problem for 'Upload!' participants

Two experiments were conducted to see how the proposed solution method behaves for instances more in line with the potential participants of 'Upload!', as described in Section 5.1.5. The results of the first experiment, the modified acceptance curves, are shown in Figure 5.9.





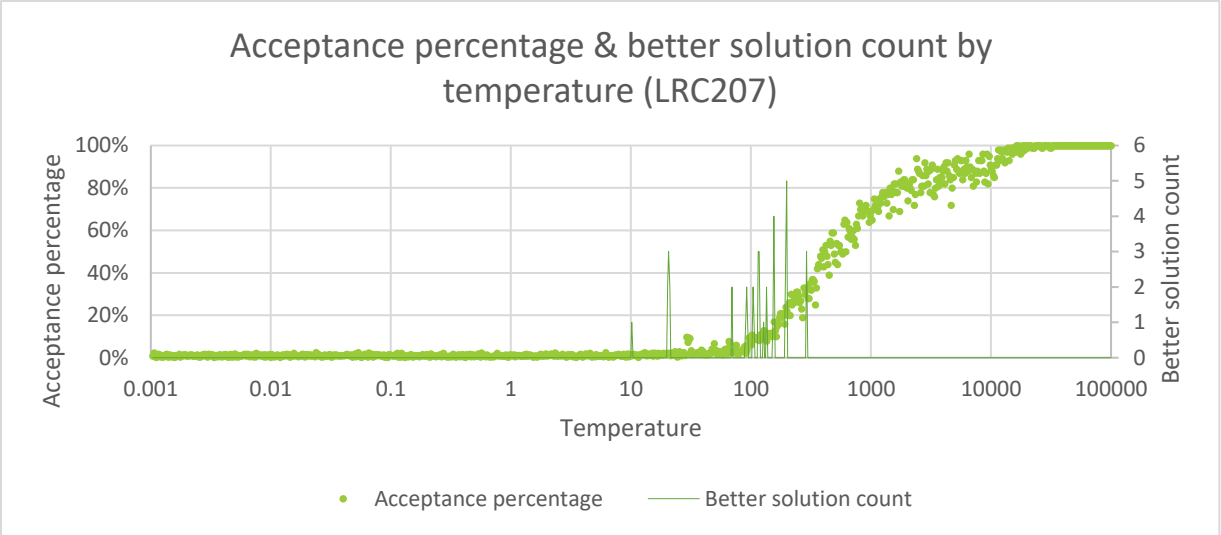


Figure 5.9 Acceptance percentage and better solution count to determine c_{start} and c_{stop} given new objective function.

When we compare the results in Figure 5.9 with the acceptance curves in Figure 5.2, we see that the acceptance percentage at high temperatures is now equal to 1. This is caused by the introduction of the cost parameter c_4, since all neighbours are now accepted because time window violations are allowed. Furthermore, we notice that, contrary to our expectations, the curves seem shifted to the right instead of to the left. In other words, at a higher temperature solutions are starting to be rejected. This seems contrary to the lower solution value of results. However, this might also be caused by the introduction of the cost parameters, since random swaps might lead to high time window violations, which corresponds with high solution values. Overall, we decide to modify the c_{start} and c_{stop} for the instances with the modified objective function to be 20,000 and 2 respectively.

The second experiment is to determine the new values for the operator probabilities p_1 through p_3. The results, both absolute and relative to the best-known solution (which is now not necessarily the best solution anymore since the objective function is changed) are shown in Figure 5.10.

Δ Solution value

		t = 120 s					t = 60 s					t = 10 s				
	p1 \ p2	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6
LC2	0,2	13,7	12,9	11,3	7,6	4,8	16,3	17,7	18,4	15,3	9,0	225,5	213,9	217,0	184	154
	0,3	12,8	9,8	4,8	2,3		19,0	16,2	17,2	15,8		209,5	193,2	131,8	66,5	
	0,4	12,4	5,2	0,3			16,2	9,0	4,5			209,5	125,1	17,6		
	0,5	9,2	4,2				51,1	9,0				97,9	33,3			
	0,6	8,7					51,3					62,1				
LRC1	0,2	52,1	22,8	22,3	27,2	18,3	34,5	32,4	26,4	26,7	30,0	82,3	64,0	59,9	59,0	49,0
	0,3	28,7	14,9	22,7	9,8		32,5	31,9	29,0	11,1		57,2	56,4	70,2	44,1	
	0,4	22,7	19,7	12,7			25,8	24,6	20,9			47,9	50,7	37,0		
	0,5	24,8	14,3				26,1	22,6				70,2	44,8			
	0,6	22,7					37,8					89,6				
LRC2	0,2	38,4	37,9	32,4	30,5	30,7	113,5	103,3	37,7	70,8	30,5	206,0	172,4	195,6	183,6	162,5
	0,3	32,5	73,2	36,8	32,9		152,4	33,6	41,9	32,8		189,2	183,6	164,1	140,0	
	0,4	37,4	33,8	19,8			100,7	47,0	15,2			162,5	162,2	143,6		
	0,5	32,9	15,8				48,1	38,2				193,6	162,2			
	0,6	31,5					41,9					153,0				

Relative Δ Solution value

		t = 120 s					t = 60 s					t = 10 s				
	p1 \ p2	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6	0,2	0,3	0,4	0,5	0,6
LC2	0,2	0,3%	0,3%	0,2%	0,1%	0,1%	0,3%	0,3%	0,4%	0,3%	0,2%	4,4%	4,2%	4,2%	3,6%	3,0%
	0,3	0,3%	0,2%	0,1%	0,0%		0,4%	0,3%	0,3%	0,3%		4,1%	3,8%	2,6%	1,3%	
	0,4	0,2%	0,1%	0,0%			0,3%	0,2%	0,1%			4,1%	2,4%	0,3%		
	0,5	0,2%	0,1%				1,0%	0,2%				1,9%	0,7%			
	0,6	0,2%					1,0%					1,2%				
LRC1	0,2	2,6%	1,1%	1,1%	1,4%	0,9%	1,7%	1,6%	1,3%	1,3%	1,5%	4,1%	3,2%	3,0%	2,9%	2,4%
	0,3	1,4%	0,7%	1,1%	0,5%		1,6%	1,6%	1,4%	0,6%		2,8%	2,8%	3,5%	2,2%	
	0,4	1,1%	1,0%	0,6%			1,3%	1,2%	1,0%			2,4%	2,5%	1,8%		
	0,5	1,2%	0,7%				1,3%	1,1%				3,5%	2,2%			
	0,6	1,1%					1,9%					4,4%				
LRC2	0,2	2,1%	2,1%	1,8%	1,7%	1,7%	6,2%	5,7%	2,1%	3,9%	1,7%	11%	9,5%	11%	10%	8,9%
	0,3	1,8%	4,0%	2,0%	1,8%		8,4%	1,8%	2,3%	1,8%		10%	10%	9,0%	7,7%	
	0,4	2,1%	1,9%	1,1%			5,5%	2,6%	0,8%			8,9%	8,9%	7,9%		
	0,5	1,8%	0,9%				2,6%	2,1%				11%	8,9%			
	0,6	1,7%					2,3%					8,4%				

Figure 5.10 Heat map (absolute and relative) of Δ solution values for different values of p_1 and p_2 and for different runtimes and benchmark groups, in case of modified objective function.

What we see in this figure is that the relative differences between the best-known solution and the calculated values are decreased a lot in comparison to the situation of the old objective function. This might be because the best-known solution is not the best solution to the problem, since we see that for some individual instances the solution value found by the algorithm is better than the best-known solution; often requiring one additional vehicle which causes the total distance to drop considerable. Furthermore, we see that the best solutions are now found using the parameters 0.4, 0.4 and 0.2 for p_1 through p_3 respectively. This indicates that the pointmove has become more important in comparison to the ordermove, which might be caused by in increased focus on distance minimisation, which can be obtained more easily using a pointmove, if the right orders are already in the right truck. We also see that the orderswap (p_3) now scores consistently lower for higher percentages (p_3 is constant over the diagonals of the Figure and the results deteriorate moving towards the left-upper corner, containing the higher p_3 values). Although not researched, this might indicate that an even lower value of p_3 scores even better.

Since the results are consistently yielding good results, and good results are already obtained for runtimes of 1 minute, we conclude that the simulated annealing method is well capable of solving these benchmark instances, especially for the objective function typically fitting the needs of targeted 'Upload!' participants. In the next chapter, we will discuss the additional steps needed to use this algorithm for implementation in an optimisation platform such as 'Upload!'.

5.2.5 Selection of inefficient trips and orders

The first result of the selection of inefficient trips is an overview of how a typical solution looks like, is shown in Figure 5.11.

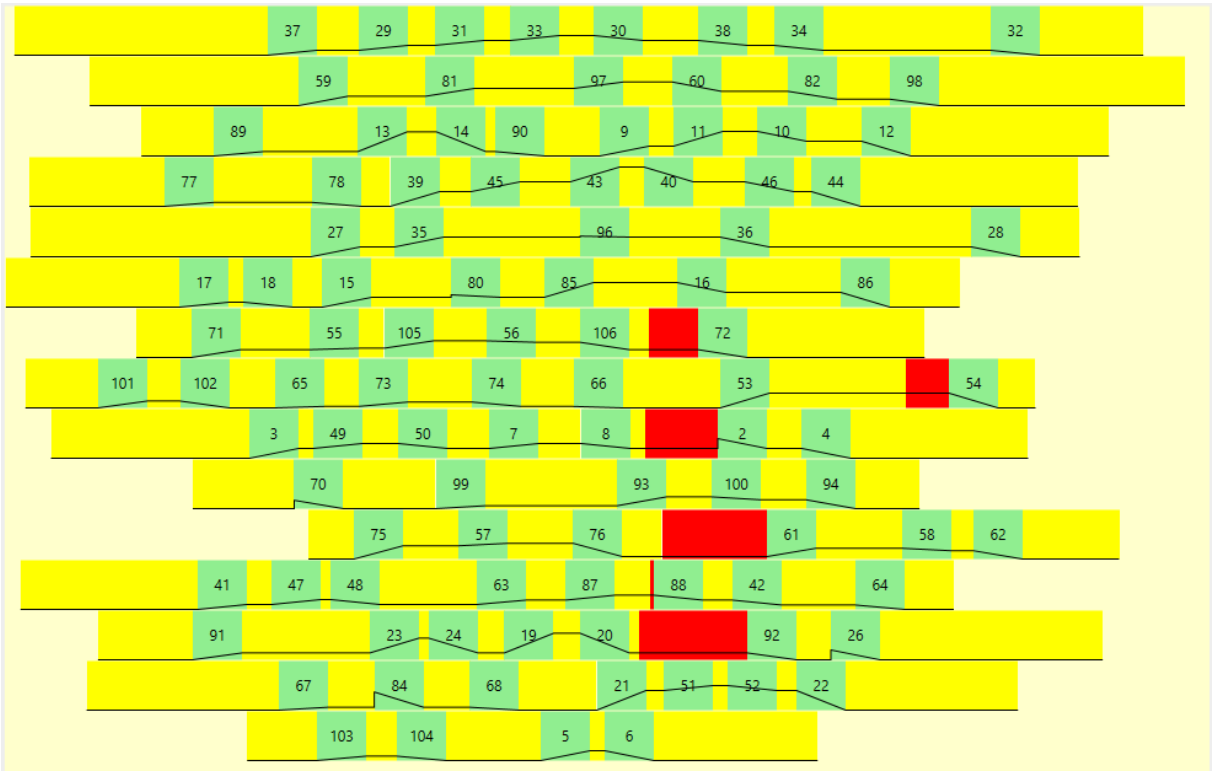


Figure 5.11 Typical trip arrangement, here for LRC101 after 60 seconds of local search using optimal operator probabilities

What we see in this figure is that the found solution for problem LRC101 consists of 15 trips, each containing between 2 (4 order points) and 4 (8 order points) orders. The picture includes the whole timeframe, i.e. left is the start time (opening of depot) and right is the latest possible arrival time at the depot (closing of depot). The green blocks are the order points visited, an uneven number corresponding with a pickup location and this number + 1 with the corresponding destination. In yellow the travel times between order points is shown and in red the waiting times are indicated. Finally the black line indicates the load factor of the truck over time. The width of the green blocks indicates the handling time at that location, which is equal for all order points. We also see that at some location the load factor takes a steep increase (for instance at location 70 and 84). This indicates that this location must be visited, but the origin and destination location is at the same position. This was introduced by Li & Lim in their benchmark to better match the Solomon's benchmark they based this benchmark on. In the next part of this research we will see that these points can have a big impact on the performance indicators.

The second experiment is to determine the relation between the four empirical PIs (relative distance, waiting time, order count and load factor. All the results found in section 5.2.4 are used, which contains 4912 trips, of which 1492 are unique. The results of this analysis are shown in Figure 5.12, where the value of the PI is shown on the y-axis and the financial PI (type c from Formula 4.3c) on the x-axis. The results are split for the different benchmark series, because different typical benchmark values are found for the different instances.

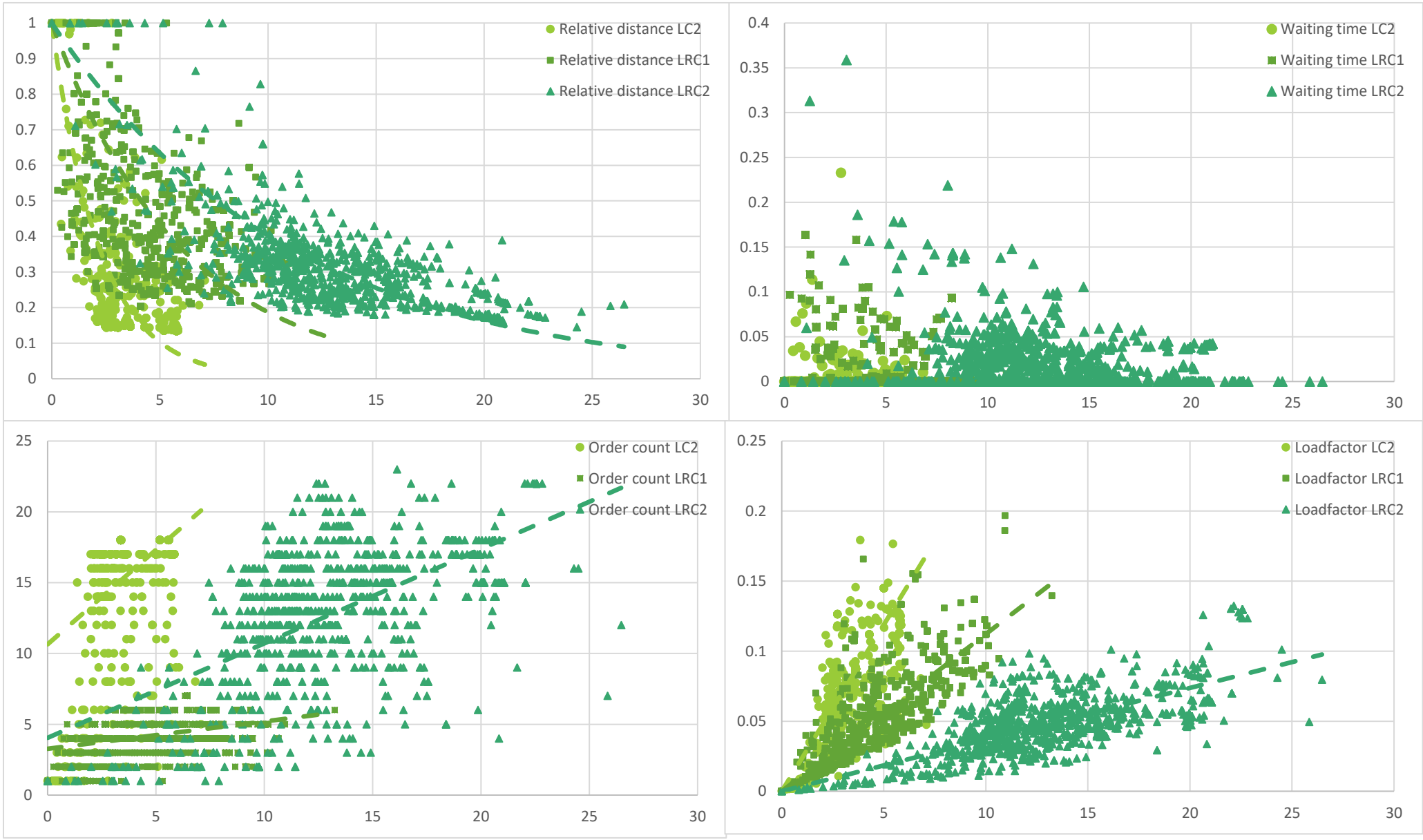


Figure 5.12 Correlation between empirical and financial PIs for different benchmark series.

What we see in these four figures is that we can found a clear relation between financial performance and relative distance, order count and load factor. This is in agreement with our expectations. We see that the best performance in terms of financial PIs is realised for the LRC2 series. When we compare the LRC2 series with the LRC1 series the LRC2 series has a higher financial performance because the order count is higher (more orders per truck means higher efficiency). When we compare the LRC2 series with the LC2 series this however the order count is rather similar. The difference here is that the distance between pickup and delivery location is much smaller and therefore the financial reward calculated using Formula 4.3c is much lower.

When we look at the relation between waiting time and financial performance, we can't discover a clear trend. This is because most inefficient trips contain only a few orders and even though these trips perform badly financially, they do not contain waiting time. In reverse, we do see that trips with a lot of waiting time typical perform not so well financially, which is as expected.

For now we mark the bottom 10% of financially bad performing trips of each benchmark series as inefficient. In Table 5.7 the characteristics of these bottom 10% trips is shown.

Table 5.7 Characteristics of bottom 10% of trips based on financial performance

		AVG	AVG	# of orders			Load factor		Rel distance	
	Count	Distance	Traveltime	AVG	MAX	# <=4	AVG	MAX	AVG	MIN
LC2	27	74,70	436,51	2,04	6	26	0,51%	2,00%	0,78	0,28
LRC1	44	142,62	220,05	2,77	5	42	1,57%	4,00%	0,58	0,34
LRC2	78	186,6	333,67	4,24	11	38	1,67%	5,74%	0,55	0,22

The most important result that we see is that the number of orders of bad performing trips is usually low, often below four orders, which is the upper bound in number of orders that can be marked as inefficient from one trip. Based on this result, the answer to the question which orders to mark as inefficient from an inefficient trip has become easy for many trips; simply take all the orders served by this trip. Testing several trips on which orders to remove from an inefficient trip containing more than 4 orders shows that removing one or more orders from such a trip will only make that trip more inefficient. In other words, the only solution to make trips more efficient is to completely remove that trip from the solution. Therefore we only take the trips with four trips or less as input for the auction.

5.2.6 Vickrey-Clarke-Groves auction

The first experiment for the VCG auction is to determine the influence of the number of participants on the algorithm runtime. The results are shown in Table 5.8.

Table 5.8 Runtimes of VCG algorithm.

M	N	|O|	Sample size	Bid-price calculation (s)	best allocation (s)	price det. (s)	Total time (s)	Avg. # exchanged	Δz
3	8	400	49	4,08	0,34	0,25	4,68	3.12	48.23
4	11	9031	37	20,8	2,51	1,88	25,2	7.32	61.97
5	13	173485	29	119	610	602	1331	10.86	87.07
6	16	6337216	2*	1866	Overflow	NA	NA		

What we see in Table 5.8 is that for an increasing number of participants (M), the number of orders than can be exchanged (N) increases linear, but the number of ways these orders can be combined

into groups of 1, 2, 3 or 4 orders each increases exponentially, following formula 5.6. As a result, the bid-price calculation and the times required determining the best allocation and the price corresponding with this best allocation also grows exponentially. The bid-price calculation time scales linear with the number of order combinations, but the time required to determine the best allocation grows again exponentially with this already exponentially growing number of order combinations. As a result, an exact algorithm as proposed functions only if the number of orders offered is low.

We also see that the higher the number of participants, the higher the fraction of orders exchanged (39%, 67% and 84%). This is as expected, since more participants means that there is a higher probability that the orders can be exchanged. We also expected the same pattern in Δz values, since more participants means more chance of good exchanges; the gain per exchanged order however remains roughly the same (6, 5.6 and 6.7) for 3, 4 and 5 participants.

The second experiment focuses on the overlap between the order points and up to which point exchange between companies remains useful. In Figure 5.13 the results of this experiment is shown. The distance to the centre point of the map is here taken in relation to the size of the grid (100x100) where all orders relate from, e.g. a distance of 1 from the centre point means that the depots are placed at the locations (-100,-100), (100,-100), (100,100) and (100,-100) relative to the centre point.

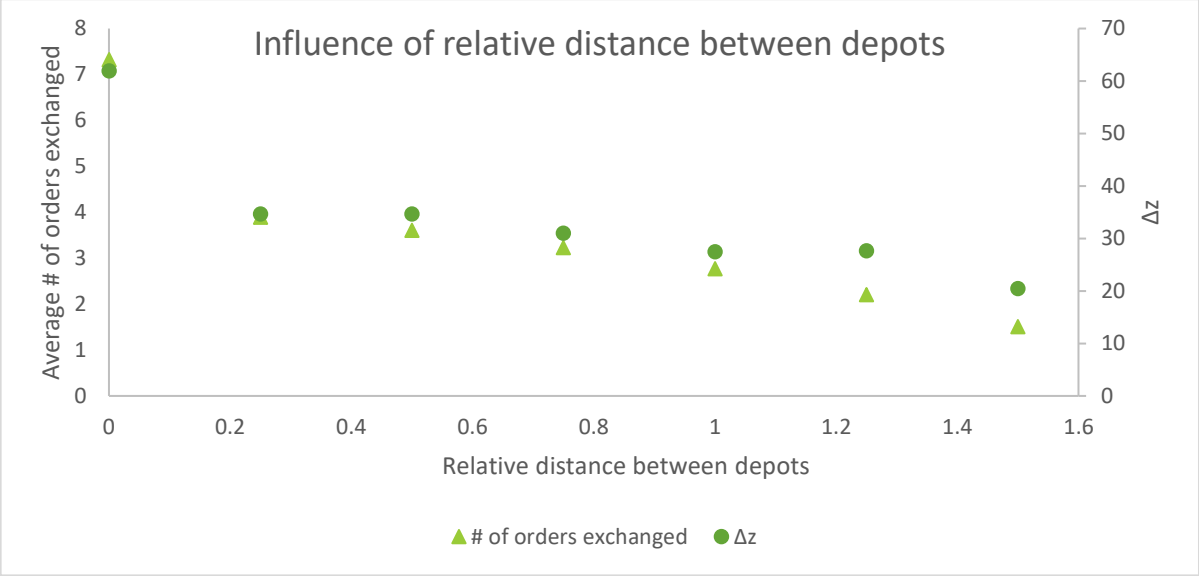


Figure 5.12 influence of distance between depots on number of orders that can be exchanged and financial gain.

In Figure 5.13 we see that the number of orders that can be exchanged drop drastically if the depots are shifted away from the centre point, indicating that the unrealistic situation where all order points are located at the same position is indeed a non-representative situation for real-life situations. If the relative distance between the depots increases, the average number of orders exchanged decreases faster than the increase in solution value, indicating that the exchange value for orders is higher. In other words, the possibility that an order can be exchanged is smaller, but if an order is exchanged the gain per order is slightly higher.

Overall we conclude that the method chosen for the exchange is not suitable for high number of participants. The results for small amounts of participants, particularly up to 5, do show that VCG auctions of inefficient trips lead to increased performance and lower costs. These results remain valid even if the distance between the depot locations of participants increase.

6 CONCLUSIONS AND RECOMMENDATIONS

In this section we summarise the conclusions of this research (Section 6.1) and provide recommendations for further research (Section 6.2).

6.1 CONCLUSIONS

The main question of this research is “How to choose, select, modify and implement models that form the algorithms of ' Upload!'? We used a context analysis, and a literature review to determine are solution approach, the chapter which answers the “choose, select and modify part” of the research question. The chapter numerical experiments then answers the implementation part of the research question. In this section we will discuss the conclusions about the Vehicle Routing Problem (6.1.1), the Selection of Inefficient orders and trips (6.1.2) and the matching algorithm (6.1.3).

6.1.1 Vehicle Routing Problem

We classified the VRP in Figure 4.1 based on a framework by Ganesh et al. (2007). Based on this classification we determined which functionality to implement in a proof of concept (Figure 4.2). Based on this overview we concluded to use simulated annealing as solving algorithm based on an initial solution found using a modified savings algorithm.

The Savings Algorithm had to be modified to incorporate the pickup and delivery character of targeted ‘Upload!’ participants. We saw that, in accordance with literature about the savings algorithm, that the solutions provided by this algorithm were not of such a quality that they can be used as final solutions. The additional constraints make the savings algorithm perform weak. We saw however that for normal runtimes (neither very short <10 seconds nor very long > 5 minutes) that the savings algorithm is a good performing starting solution for the simulated annealing algorithm.

The Simulated Annealing algorithm was parameterized to find both the best solutions to compare with the benchmark as well as for a typical targeted ‘Upload!’ user. The methodology described can also be executed if the algorithm is used for real participants. The more initial solutions there are, the better the parameterization can be done. This also means that the algorithms performance can be optimized using a feedback loop.

We saw that for the optimal parameterization, which includes temporarily accepting infeasible (either because of capacity or because of time window violation) trips, the algorithm can find near optimal solutions to the pickup and delivery VRP with time windows in about 2 minutes for instances with 50 pickup and delivery locations. The addition of Large Neighbourhood Search did not improve the solutions due to the long runtime of this algorithm. We suspect that by making the LNS less time consuming i.e. better implemented, LNS can significantly further increase the quality of the VRP solutions.

6.1.2 Selection of Inefficient orders and trips

Using a stakeholders analyse we determined what the characteristics of inefficient trips are. We determined four empirical performance indicators (PIs); namely the load factor, the waiting time, the number of orders and the relative distance driven. We compared these empirical PIs with a financial performance indicator comparing the costs of the trip as defined in the objective function with the expected revenue of the trip, which we set equal to a fixed amount per unit distance per measure of load driven.

We found for that the relative distance, the order count and the load factor clearly correlates with the financial performance of the trip. The waiting time correlates with the financial performance one way; high waiting time lead to low financial performance, otherwise is low financial performance not a measure for high waiting time.

The bottom 10 percentage performing trips are largely caused by trips containing 4 or less orders, were 4 is also the upper boundary for the number of orders from one trip that can be marked as inefficient, due to computational restrictions for the matching algorithm. As such, we concluded that for the benchmark series we only mark orders containing of 4 or less orders inefficient. Since the individual performance of trips only decreases when inefficient trips are removed from bad performing trips, all orders of these trips are marked as inefficient and offered on the market place.

6.1.3 Matching Algorithm

A successful matching algorithm must comply with three requirements; it must be a combinatorial auction, truthful bidding must be the optimal bidding strategy and every participant just have to place a single bid on every (combination of) orders they valuate. The Vickrey-Clarke-Groves (VCG) auction is designed for these requirements and as such selected as matching algorithm.

Numerical experiments show that the exact implementation as proposed in this experiment can handle instances of 11 orders from 4 participants in reasonable time. Moving up from 4 to 5 participants increases the runtime from about half a minute total to over 2 hours since the number of possible order combinations increases exponentially.

6.2 Recommendations for further research

During this research, there were certain factors that could not be researched or implemented due to time restrictions. These possibilities are listed here as recommendations for further research.

Additional VRP constraints

We made a selection of actual constraints to simplify the implementation of the VRP algorithms. In order to create an algorithm capable of solving the VRP of future ‘Upload!’ participants, additional constraints must be implemented.

Improved Large Neighbourhood Search

Large Neighbourhood search (LNS) shows promising results both in literature as well as in the experiments done in this research. Because the way the algorithm was implemented however, LNS was outperformed on speed by the standard local search operators. With an improved implementation we believe that LNS can be a valuable addition to the existing algorithm.

Inefficient trip feedback algorithm

Based on the benchmark no clear cut-off values determining when a trip is inefficient could be determined. We would propose to create an additional algorithm which uses a feedback loop to determine per ‘Upload!’ participants which orders cause inefficiency. The feedback loop ensures that the algorithm remains performing optimal, even when the companies characteristics are evolving over time.

VCG Heuristic

The exact algorithm implemented to find the best allocation of orders as a result of the Vickrey-Clarke-Groves auction is incapable of finding these allocations for larger than 4 participants. Therefore the exact approach is not feasible for implementation within ‘Upload!’, given that ‘Upload!’ targets much more than four participants. Even when code optimization is performed, the

exact algorithm will still require too much computation time. A heuristic determining a good allocation of orders between participants is therefore required. Heuristic might include the same pattern as used for the VRP, e.g., a greedy construction algorithm coupled with a local search algorithm to optimise.

7 REFERENCES

Agarwal, Richa, & Ergun, Özlem. (2010). Network Design and Allocation Mechanisms for Carrier Alliances in Liner Shipping. *Operations Research, 58*(6), 1726-1742. doi: doi:10.1287/opre.1100.0848

Anbuudayasankar, S.P., Ganesh, K., & Mohapatra, S. (2014). *Models for Practical Routing Problems in Logistics* (1 ed.). Switzerland: Springer International Publishing.

Berger, Susanne, & Bierwirth, Christian. (2010). Solutions to the request reassignment problem in collaborative carrier networks. *Transportation Research Part E: Logistics and Transportation Review, 46*(5), 627-638. doi: http://dx.doi.org/10.1016/j.tre.2009.12.006

Brotcorne, L., Laporte, G., & Semet, F. (2002). Fast heuristics for large scale covering-location problems. *Computers & Operations Research, 29*(6), 651-665. doi: http://dx.doi.org/10.1016/S0305-0548(99)00088-X

Clifton, Chris, Iyer, Ananth, Cho, Richard, Jiang, Wei, Kantarcıoğlu, Murat, & Vaidya, Jaideep. (2008). An Approach to Securely Identifying Beneficial Collaboration in Decentralized Logistics Systems. *Manufacturing & Service Operations Management, 10*(1), 108-125. doi: doi:10.1287/msom.1070.0167

Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society, 53*(5), 512-522. doi: 10.1057/palgrave/jors/2601319

Cordeau, J.F., Laporte, G., Savelsbergh, M.W.P., & Vigo, D. (2006). Vehicle routing. *Transportation, handbooks in operations research and management science, 14*, 367-428.

Cruijssen, Frans, Bräysy, Olli, Dullaert, Wout, Fleuren, Hein, & Salomon, Marc. (2007). Joint route planning under varying market conditions. *International Journal of Physical Distribution & Logistics Management, 37*(4), 287-304. doi: doi:10.1108/09600030710752514

Dai, Bo, & Chen, Haoxun. (2011). A multi-agent and auction-based framework and approach for carrier collaboration. *Logistics Research, 3*(2-3), 101-120. doi: 10.1007/s12159-011-0046-9

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

Eilon, S., Watson-Gandy, C.D.T, & Christofides, N. (1971). *Distribution management*: Griffin London.

Fischer, Klaus, Mu¨ller, Jo¨RG P., & Pischel, Markus. (1996). Cooperative transportation scheduling: An application domain for dai. *Applied Artificial Intelligence, 10*(1), 1-34. doi: 10.1080/088395196118669

Fisher, M. (1995). Chapter 1 Vehicle routing *Handbooks in Operations Research and Management Science* (Vol. Volume 8, pp. 1-33): Elsevier.

Ganesh, K., Sam Nallathambi, A., & Narendran, T.T. (2007). Variants, solution approaches and applications for Vehicle Routing Problems in supply chain: agile framework and comprehensive review. *International Journal of Agile Systems and Management, 2*(1), 50-75. doi: doi:10.1504/IJASM.2007.015681

Garey, M.R., & Johnson, D.S. (1979). Computers and intractability: a guide to NP-completeness: WH Freeman New York.

Gendreau, Michel, Potvin, Jean-Yves, Bräumlaysy, Olli, Hasle, Geir, & Løkketangen, Arne. (2008). Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography. In B. Golden, S. Raghavan & E. Wasil (Eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (Vol. 43, pp. 143-169): Springer US.

Heerkens, H. (2005). Algemene Bedrijfskundige Aanpak. *Enschede, Nederland*.

Hernández, Salvador, Peeta, Srinivas, & Kalafatas, George. (2011). A less-than-truckload carrier collaboration planning problem under dynamic capacities. *Transportation Research Part E: Logistics and Transportation Review, 47*(6), 933-946. doi: http://dx.doi.org/10.1016/j.tre.2011.03.001

Ho, W., Ho, G.T.S., Ji, P., & Lau, H.C.W. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence, 21*(4), 548-557. doi: http://dx.doi.org/10.1016/j.engappai.2007.06.001

Houghtalen, Lori, Ergun, Özlem, & Sokol, Joel. (2011). Designing Mechanisms for the Management of Carrier Alliances. *Transportation Science, 45*(4), 465-482. doi: doi:10.1287/trsc.1100.0358

Joubert, JW, & Claasen, SJ. (2006). A sequential insertion heuristic for the initial solution to a constrained vehicle routing problem. *ORiON: The Journal of ORSSA, 22*(1), 105-116.

Krajewska, MartaAnna, & Kopfer, Herbert. (2006). Collaborating freight forwarding enterprises. *OR Spectrum, 28*(3), 301-317. doi: 10.1007/s00291-005-0031-2

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59*(3), 345-358. doi: 10.1016/0377-2217(92)90192-C

Mitchell, Ronald K, Agle, Bradley R, & Wood, Donna J. (1997). Toward a theory of stakeholder identification and salience: Defining the principle of who and what really counts. *Academy of management review, 22*(4), 853-886.

Montoya-Torres, J.R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers and Industrial Engineering, 79*, 115-129. doi: 10.1016/j.cie.2014.10.029

Morash, Edward A., Droge, Cornelia L. M., & Vickery, Shawnee K. (1996). Strategic logistics capabilities for competitive advantage and firm success. *Journal of Business Logistics, 17*(1), 1-22.

Özener, Okan Örsan, Ergun, Özlem, & Savelsbergh, Martin. (2011). Lane-Exchange Mechanisms for Truckload Carrier Collaboration. *Transportation Science, 45*(1), 1-17. doi: doi:10.1287/trsc.1100.0327

Poot, A., Kant, G., & Wagelmans, A.P.M. (2002). A savings based method for real-life vehicle routing problems. *Journal of the Operational Research Society, 53*(1), 57-68.

Regan, Amelia, & Song, Jiongjiong. (2003). An auction based collaborative carrier network. *Transportation Research Part E: Logistics and Transportation Review, 2*.

Schwind, Michael, Gujo, Oleg, & Vykoukal, Jens. (2009). A combinatorial intra-enterprise exchange for logistics services. *Information Systems and e-Business Management, 7*(4), 447-471. doi: 10.1007/s10257-008-0102-4

Simatupang, Togar M, & Sridharan, Ramaswami. (2002). The collaborative supply chain. *The International Journal of Logistics Management, 13*(1), 15-30.

Sprenger, Ralf, & Mönch, Lars. (2012). A methodology to solve large-scale cooperative transportation planning problems. *European Journal of Operational Research, 223*(3), 626-636. doi: http://dx.doi.org/10.1016/j.ejor.2012.07.021

Toth, P., & Vigo, D. (2001). *The vehicle routing problem*: Society for Industrial and Applied Mathematics.

Verdonck, Lotte, Caris, A. N., Ramaekers, Katrien, & Janssens, Gerrit K. (2013). Collaborative Logistics from the Perspective of Road Transportation Companies. *Transport Reviews, 33*(6), 700-719. doi: 10.1080/01441647.2013.853706

VerkeerWaterstaat, Inspectie. (2010). Rij- en rusttijden vrachauto en touringcar. https://www.ilent.nl/Images/Infoblad%20Rij-%20en%20rusttijden%20vrachtauto%20en%20touringcar%20obv%20Vo%20561_tcm334-318004.pdf

Weise, T., Podlich, A., & Gorldt, C. (2009). Solving Real-World Vehicle Routing Problems with Evolutionary Algorithms. In R. Chiong & S. Dhakal (Eds.), *Natural Intelligence for Scheduling, Planning and Packing Problems* (Vol. 250, pp. 29-53): Springer Berlin Heidelberg.

Zhou, Guanghui, Hui, Yer Van, & Liang, Liang. (2011). Strategic alliance in freight consolidation. *Transportation Research Part E: Logistics and Transportation Review, 47*(1), 18-29. doi: http://dx.doi.org/10.1016/j.tre.2010.07.002

APPENDIX A – BEST RESULTS OF LI LIM BENCHMARK

Table A.1 Best known results for relevant instances of the Li Lim Benchmark

Instance	Vehicles	Distance	Time	Reference	Date
lc201	3	591,56	9591,56	Li & Lim	2001
lc202	3	591,56	9591,56	Li & Lim	2001
lc203	3	591,17	9591,17	BVH	27-6-2003
lc204	3	590,6	9590,6	SAM::OPT	11-4-2003
lc205	3	588,88	9588,88	Li & Lim	2001
lc206	3	588,49	9588,49	Li & Lim	2001
lc207	3	588,29	9588,29	Li & Lim	2001
lc208	3	588,32	9588,32	BVH	27-6-2003
lrc101	14	1708,8	2767,13	Li & Lim	2001
lrc102	12	1558,07	2586,73	SAM::OPT	19-2-2003
lrc103	11	1258,74	2312,96	Li & Lim	2001
lrc104	10	1128,4	2139,37	Li & Lim	2001
lrc105	13	1637,62	2683,38	Li & Lim	2001
lrc106	11	1424,73	2424,73	SAM::OPT	28-2-2003
lrc107	11	1230,14	2230,14	SAM::OPT	18-2-2003
lrc108	10	1147,43	2147,43	SAM::OPT	28-2-2003
lrc201	4	1406,94	3062,68	SAM::OPT	28-2-2003
lrc202	3	1374,27	2519,55	Li & Lim	2001
lrc203	3	1089,07	2529,78	Li & Lim	2001
lrc204	3	818,66	2184,51	SAM::OPT	23-3-2003
lrc205	4	1302,2	3085,52	Li & Lim	2001
lrc206	3	1159,03	2255,81	SAM::OPT	12-3-2003
lrc207	3	1062,05	2065,72	SAM::OPT	4-1-2004
lrc208	3	852,76	1852,76	Li & Lim	2001

References:

BVH - Bent, R. and Van Hentenryck. P. A: Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows. In Principles and Practice of Constraint Programming (2003).

Li & Lim - Li H. and A. Lim: A MetaHeuristic for the Pickup and Delivery Problem with Time Windows, In Proceedings of the 13th International Conference on Tools with Artificial Intelligence, Dallas, TX, USA, 2001.

SAM::OPT - Hasle G., O. Kloster: Industrial Vehicle Routing Problems. Chapter in Hasle G., K-A Lie, E. Quak (eds): Geometric Modelling, Numerical Simulation, and Optimization. ISBN 978-3-540-68782-5, Springer 2007.

Results and detailed solutions adapted from: Transportation Optimization Portal of SINTEF Applied Mathematics at http://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/100-customers/