Collaborative Gain Assessment for the Dynamic Multi-Objective Vehicle Routing Problem

Luke Borst
Collaborative Gain Assessment for the Dynamic Multi-Objective Vehicle Routing Problem

by

Luke Borst

in fulfilment to obtain the degree of Master of Science at the Delft University of Technology,

Student number: 4313755
Project duration: April, 2019 – November, 2019
Supervisors: B. Atasoy, TU Delft
N. Maleki, Quicargo
R.R. Negenborn, TU Delft
Abstract

Inefficiency in trucking has both economical and ecological consequences. In literature, horizontal collaboration is identified as one of the most promising solutions to this problem. However, collaboration is not often encountered in trucking industry, since carriers are hesitant to join collaborations. This is caused by a fear of sharing information and an uneven distribution of requests between carriers. Furthermore, the amount of collaborative gain remains still unclear when dealing with a dynamically changing environment, which is often present.

This work aims to fill this knowledge gap. Firstly by developing a vehicle routing problem (VRP) model, which includes dynamic requests, horizontal collaboration and customer satisfaction. Next to that, a solution method is developed which is based on Adaptive Large Neighborhood Search (ALNS). The performance of the implemented method is evaluated with a computational study of 10 instances. The performance is compared with a full information (FI) method and another dynamic method (SI). The FI method evaluates the same instances, with all information known beforehand, thereby acting as a lower bound. The newly developed solution method has shown to perform 39% worse compared to full information, while being able to deal with dynamic requests. The SI method shows an increase of 60% in routing costs when compared with FI.

Next the development of the model and solution method, the implemented method is used to investigate the collaborative gain for four levels of collaboration in a dynamic environment. The level of collaboration is expressed by the number of requests that is reassigned between carriers. The results show that up to 23% collaborative gain can be realised in the dynamic setting.
This work is the result of the graduation project as final part of the master Transport Engineering and Logistics. First of all I would like to thank Quicargo, who have given the opportunity to me to conduct this work together with them. My colleagues at Quicargo made my graduation project so much more enjoyable. I specifically want to thank Nima Maleki, who as my company supervisor was always able to help me with the problems that I was facing.

Also I would like to thank my TU Delft supervisors. First of all Bilge Atasoy, for the numerous discussions we had, and allowing me to steer the research in a direction that I thought was interesting. As chair of the committee, Rudy Negenborn has provided me with a fresh view during the meetings that we had.

Lastly, a word of thanks for my friends, who have allowed me to escape the grind of the graduation process when needed, and my parents, for the unconditional support they gave me during my time in Delft.

Luke Borst
Delft, November 2019
<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALNS</strong></td>
<td>Adaptive Large Neighborhood Search</td>
</tr>
<tr>
<td><strong>CGA</strong></td>
<td>Collaborative Gain Assessment</td>
</tr>
<tr>
<td><strong>DARP</strong></td>
<td>Dial A Ride Problem</td>
</tr>
<tr>
<td><strong>DI</strong></td>
<td>Destroy &amp; Insertion method</td>
</tr>
<tr>
<td><strong>DVRP</strong></td>
<td>Dynamic Vehicle Routing Problem</td>
</tr>
<tr>
<td><strong>FI</strong></td>
<td>Full Information method</td>
</tr>
<tr>
<td><strong>GA</strong></td>
<td>Genetics Algorithm</td>
</tr>
<tr>
<td><strong>GIL</strong></td>
<td>Global Interpreter Lock</td>
</tr>
<tr>
<td><strong>KPI</strong></td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td><strong>LNS</strong></td>
<td>Large Neighborhood Search</td>
</tr>
<tr>
<td><strong>OSRM</strong></td>
<td>Open Source Routing Machine</td>
</tr>
<tr>
<td><strong>PDP</strong></td>
<td>Pick-up and Delivery Problem</td>
</tr>
<tr>
<td><strong>RRT</strong></td>
<td>Record to Record Travel</td>
</tr>
<tr>
<td><strong>SA</strong></td>
<td>Simulated Annealing</td>
</tr>
<tr>
<td><strong>SI</strong></td>
<td>Simple Insertion method</td>
</tr>
<tr>
<td><strong>TA</strong></td>
<td>Threshold Acceptance</td>
</tr>
<tr>
<td><strong>TSP</strong></td>
<td>Traveling Salesman Problem</td>
</tr>
<tr>
<td><strong>VRPP</strong></td>
<td>Vehicle Routing Problem with Profits</td>
</tr>
<tr>
<td><strong>VRPTW</strong></td>
<td>Vehicle Routing Problem with Time Windows</td>
</tr>
<tr>
<td><strong>VRP</strong></td>
<td>Vehicle Routing Problem</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Empty road journeys by type of operation per country. Percentual share in in vehicle-kilometers. [11] ....................................................... 2
1.2 Overview of the Quicargo platform, showing the optimal insertion of requests, performed by the so called matching algorithm [39] .............................. 4

2.1 Example of DVRP [37] ................................................................. 6
2.2 Vertical (1,5) and horizontal (2,4) collaboration with company (3) in a supply chain [1] .......................................................... 8
2.3 Example of varying geographic distribution levels of requests. O1 being clustered around the depot, O3 being randomly distributed. [15] ............... 9
2.4 Service level function of soft time window [47] .................................. 11
2.5 Euler diagram for computational complexity [53] .............................. 11
2.6 Overview of local search operators [3] ........................................... 13

3.1 Overview of dynamic problem. Time $T$ has two influences on the static problem as presented in equations (3.1). Additional requests appear via the platform (set $D$), and part of the solution is fixed (set $N_f$). The new solution is provided to the carriers and to the solver for the next iteration ................................. 19

4.1 Overview of solution steps [34] ..................................................... 23
4.2 Validation results: 4.2a shows the relative number of routes with respect to the best known solution. 4.2b shows the relative distance with respect to the best known solution. ............................................................... 30

5.1 Overview of request locations for instance F. Left side shows the initial requests. Right side shows the dynamic requests. The depot locations are encircled in black 35
5.2 Time window length and load quantity for all requests. Loading rate is shown as fraction of vehicle capacity ...................................................... 35
5.3 Time stamp and time gap for all dynamic requests. Time stamp is the time of receival of the requests, Time gap is the gap between the time stamp and the end of the time window .................................................. 36
5.4 Routing costs for all instances at varying number of initial iterations using FI method ................................................................. 38
5.5 Routing costs for all instances at varying simulation speeds using DI method 38
5.6 Fraction of rerouted requests, compared with the initial number of requests. 39
5.7 Normalized fraction of rerouted requests, compared with the full collaborative case ................................................................. 40
5.8 Fraction of the total routing costs from the collaborative penalty .......... 40
5.9 Routing costs of three strategies for the small instance at varying collaborative levels ................................................................. 41
5.10 Routing costs of three strategies for instance E at varying collaborative levels 42
5.11 Routing costs of three strategies for instance G at varying collaborative levels 42
5.12 Computational times for the strategies full information (FI), destroy & insertion (DI), and simple insertion (SI) ........................................... 43
5.13 Averaged collaborative gain assessment for the three strategies full information (FI), destroy & insertion (DI), and simple insertion (SI) ............... 43
5.14 Averaged number of reassigned requests for the three strategies full information (FI), destroy & insertion (DI), and simple insertion (SI). The number of reassignments is a fraction of the total number of requests 46
5.15 Histogram of customer inactivity risk .......................... 46
5.16 Routing costs for all instances excluding and including the customer satisfaction 47

B.1 Overview of request locations for instance A. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 62
B.2 Overview of request locations for instance B. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 62
B.3 Overview of request locations for instance C. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 63
B.4 Overview of request locations for instance D. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 63
B.5 Overview of request locations for instance E. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 64
B.6 Overview of request locations for instance F. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 64
B.7 Overview of request locations for instance G. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 65
B.8 Overview of request locations for instance H. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 65
B.9 Overview of request locations for instance I. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 66
B.10 Overview of request locations for instance J. Left side shows the initial requests. Right side shows the dynamic requests. ........................................ 66

C.1 Routing costs of three strategies for instance A at varying collaborative levels .. 67
C.2 Routing costs of three strategies for instance B at varying collaborative levels .. 67
C.3 Routing costs of three strategies for instance C at varying collaborative levels .. 68
C.4 Routing costs of three strategies for instance D at varying collaborative levels .. 68
C.5 Routing costs of three strategies for instance D at varying collaborative levels .. 68
C.6 Routing costs of three strategies for instance F at varying collaborative levels .. 68
C.7 Routing costs of three strategies for instance G at varying collaborative levels .. 69
C.8 Routing costs of three strategies for instance H at varying collaborative levels .. 69
C.9 Routing costs of three strategies for instance I at varying collaborative levels .. 70
C.10 Routing costs of three strategies for instance J at varying collaborative levels .. 70
# List of Tables

3.1 Overview of symbols used in problem formulation ........................................ 18
4.1 Description of weight adjustment parameters .................................................. 24
5.1 Overview of request data .................................................................................. 34
5.2 Request attributes ............................................................................................. 34
5.3 Vehicle attributes .............................................................................................. 36
5.4 Overview of algorithm parameters .................................................................... 37
5.5 Competitive ratios for the small scale instance (S) at all collaborative levels. Ratios for both the destroy & insertion method (DI), and the simple insertion method (SI) are determined with respect to the full information (FI) method .............................................................. 41
5.6 Overview of performed experiments ................................................................... 42
5.7 Competitive ratios for all instances at all collaborative levels. Ratios for both the destroy & insertion method (DI), and the simple insertion method (SI) are determined with respect to the full information (FI) method. Per column, the minimum and maximum are highlighted in bold. Note that highlighting is based on unrounded values ........................................................................................................ 43
5.8 Collaborative gain for each method; Full information (FI), destroy & insertion (DI), and simple insertion (SI) and three levels of collaboration. For each method, each level is compared with the non collaborative case $\xi = 1$. Per column, the minimum and maximum are highlighted in bold. Note that highlighting is based on unrounded values ........................................................................................................ 44
5.9 Fraction of number of reassignments for each method; Full information (FI), destroy & insertion (DI), and simple insertion (SI) and three levels of collaboration. For each method, each level is compared with the non collaborative case $\xi = 0$. Per column, the minimum and maximum are highlighted in bold. Note that highlighting is based on unrounded values ........................................................................................................ 45
A.1 Validation results for instance set C1 ................................................................. 58
A.2 Validation results for instance set C2 ................................................................. 58
A.3 Validation results for instance set R1 ................................................................. 58
A.4 Validation results for instance set R2 ................................................................. 59
A.5 Validation results for instance set RC1 .............................................................. 59
A.6 Validation results for instance set RC2 .............................................................. 59
# Contents

List of Figures ix
List of Tables xi

## 1 Introduction
1.1 Background ........................................ 1
1.2 Research motivation ................................ 3
1.3 Scope ............................................. 3
1.4 Research questions ................................ 4
1.5 Report structure .................................. 4

## 2 Literature review
2.1 Problem introduction ................................ 5
2.2 Dynamic vehicle routing problem .................. 6
    2.2.1 Degree of dynamism ............................ 6
    2.2.2 Consequences for solution methods .......... 7
2.3 Collaboration ....................................... 8
    2.3.1 Collaboration in road based transport ..... 8
    2.3.2 Centralized planner ........................... 9
    2.3.3 Decentralized planner ....................... 9
    2.3.4 Collaboration fairness ...................... 10
2.4 Service levels ..................................... 10
    2.4.1 Vehicle routing problem with profits .... 10
    2.4.2 Time window relaxation .................... 10
2.5 Solution methods .................................. 10
    2.5.1 Computational complexity ................... 11
2.6 Exact methods .................................... 11
2.7 Heuristics ........................................ 12
    2.7.1 Local search ................................ 12
    2.7.2 Large neighborhood search ............... 12
2.8 Metaheuristics .................................... 13
    2.8.1 Simulated annealing .......................... 13
    2.8.2 Tabu search ................................ 13
    2.8.3 Genetic algorithm ............................ 14
    2.8.4 Other ........................................ 14
2.9 Concluding remarks ................................ 14

## 3 Problem formulation
3.1 Problem formulation ............................... 17
    3.1.1 Static problem ................................ 17
    3.1.2 Dynamic problem ............................. 19
3.2 Horizontal collaboration ........................... 19
3.3 Customer satisfaction ................................ 20
3.4 Including conflicting objectives .................. 20
3.5 Concluding remarks ................................ 21

## 4 Solution method
4.1 General framework ................................. 23
4.2 Initial solution .................................... 24
4.3 Adaptive Large Neighborhood Search .............. 24
    4.3.1 Roulette wheel selection .................... 24
4.3.2 Adaptive weight adjustment ........................................... 24
4.3.3 Destroy heuristics ......................................................... 26
4.3.4 Insertion heuristics ....................................................... 27
4.4 Solution acceptance .......................................................... 29
4.5 Validation .............................................................................
  4.5.1 Route minimization ....................................................... 30
  4.5.2 Results ............................................................................ 30
4.6 Solution methods ............................................................... 31
4.7 Concluding remarks .......................................................... 31

5 Experimental evaluation ......................................................... 33
  5.1 Input data ............................................................................ 33
    5.1.1 Requests ........................................................................ 33
    5.1.2 Carriers: depot and initial schedule .................................. 35
    5.1.3 Vehicles ......................................................................... 36
  5.2 Experimental setup ............................................................ 36
    5.2.1 Software implementation ................................................. 36
    5.2.2 Hardware ....................................................................... 37
    5.2.3 Algorithm parameters .................................................... 37
    5.2.4 Small scale experiment ................................................ 40
  5.3 Results ................................................................................. 41
    5.3.1 Method comparison: performance .................................... 41
    5.3.2 Method comparison: computational times ......................... 43
    5.3.3 Collaborative gain: routing cost ....................................... 44
    5.3.4 Collaborative gain: reassignments .................................... 45
    5.3.5 Including customer satisfaction ....................................... 45
  5.4 Concluding remarks .......................................................... 47

6 Conclusion .............................................................................. 49
  6.1 Discussion ........................................................................... 50
  6.2 Future research ................................................................. 50

Bibliography ............................................................................. 53
A Validation results ..................................................................... 57
B Geographical request distribution ............................................. 61
C Results of all instances ........................................................... 67
Introduction

1.1. Background

When travelling between two major cities in the Netherlands during rush hour, one can notice immediately how crowded the Dutch road system can be. Next to people travelling to and from their jobs every day, the rightmost lane is often filled with trucks doing their daily pickups and deliveries throughout the country. When one starts to investigate what these trucks are transporting, it is discovered that over 25% of these trucks is empty, as shown in figure 1.1. This inefficiency in road transport has both economical and ecological consequences.

Firstly, for multiple years, the transport sector has reported that revenues are growing larger, while the growth of profit is lacking. The effect of the decreasing profit margins puts pressure on the transport sector [48]. The sector reports that the increased costs of labour and fuel are the main cause of decreasing profit margins. Increasing transport efficiency may directly benefit the profit margins.

On top of decreasing profit margins for the transport companies, transport inefficiency result in an increased pressure on the road network. Increased pressure on the road network leads to an increased number of traffic jams and accidents. Therefore, keeping the amount of traffic manageable is preferred. In the Netherlands, the economical consequences of traffic jams accounted for a loss of 1.3 billion euros last year [40]. Next to that, the Dutch government spends yearly around 2 to 3 billion euros in order to maintain and improve its road network.

Next to economical impact, the transport sector is a major contributor to the emission of greenhouse gasses. For example, in 2017 more than 5 million kilograms of CO$_2$ was emitted by road-based transport [4]. With a demand for more sustainable usage of our planet’s resources, the demand for efficient transport is growing.

Parallel to the identified issues with inefficient shipping, technological opportunities from different sectors may prove beneficial, when applied to the road transport sector. In the last decade, multiple tech companies have incorporated the concept of the sharing economy. Examples are Airbnb and Blablacar, where you can offer unoccupied rooms in your house, or unused seats of your car trip, via an online platform. These disruptive technologies have shown to be highly attractive [24] [23]. By extending this sharing concept to road transport there is a great potential for improvement.

The identification of, on the one hand the negative effects of empty haulage, and the mentioned technological opportunities on the other, leads to the conclusion that there is an opportunity to seize. Increasing shipping efficiency may provide multiple benefits. Firstly, an increase in the utilization rate of trucks will have a direct benefit regarding the operational costs of a transport company. Secondly, increased transport efficiency will not only benefit the transport companies directly, but will, by decreasing road network pressure, have beneficial economical effects on a larger scale as well.
1. Introduction

Figure 1.1: Empty road journeys by type of operation per country. Percentual share in in vehicle-kilometers. [11]
1.2. Research motivation

The previously mentioned opportunity of increasing trucking efficiency is already being commercialized by the company Quicargo, with whom this work is executed together with. Currently, they are providing an online platform for shippers and carriers, that tries to match empty space in carrier’s trucks with requests provided by shippers. Since Quicargo does not have trucks themselves, they rely on carriers that are connected with the platform to process the requests that they receive via the platform.

Quicargo has identified two ways to increase the utilization rate of the carrier’s trucks. Firstly, by placing their own orders at the optimal location in a carrier’s schedule, called insertion, as shown in figure 1.2. Secondly, an additional improvement can be realized when the platform owner is allowed to reassign requests between carriers, even if those requests were not received via the platform. In the case that requests initially belonging to one of the other carriers are reassigned, we speak of collaboration.

A simple example of collaborative gain: Assume that two carriers, both operating from the same city, have to deliver half a truck load of goods to the same company. It is the only task that they have for that day. In that case, the two trucks are both traveling the same route, with only half of their capacity used. When collaboration is applied, the platform owner can identify this inefficiency, and put both requests in a single truck.

This example also shows one of the negative effects of collaboration: since only a single truck is in use now, the other carrier is out of work. Although the overall efficiency is higher, one carrier is unlikely to accept this. The fear of losing work and customers makes carriers hesitant to join collaborations. Carriers hesitant to join collaborations is a major reason why collaboration is not often seen in the transport industry.

The goal of this work is to provide an algorithm that increases the routing efficiency of trucks. This will be done in a dynamic and collaborative environment. The dynamic environment will reflect the real life case in which the carriers receive additional requests. Additionally, the collaborative aspect will hopefully have an extra effect on the transport efficiency. The transport efficiency will be reflected by means of transport costs as the main KPI.

Next to the development of a model and solution method, this work investigates the added value of collaboration in a dynamic environment. For static cases, where the number of requests is fixed during the day, collaboration has already shown to be beneficial. For the dynamic case this is still unclear. The results of this work may convince carriers to join a collaborative network, since the collaborative gain is significant, even with limited levels of collaboration.

1.3. Scope

The work is bounded by a scope for several reasons. Firstly, it is in place so that the research does not continue indefinitely. Secondly, by limiting the scope of the project, one aspect of the problem can be investigated thoroughly.

Hereby, this work will only focus on the actual routing of trucks between the provided requests. The set of initial requests is completely known beforehand, and additional requests will be revealed throughout the day.

Secondly, the pricing of requests in not considered, but only the routing costs. The carrier’s pricing strategies are not well known, and therefore excluded. When requests are reassigned between carriers, not only should the request be rerouted, but the gain that is obtained should be shared fairly between carriers. The investigation of these profit sharing mechanisms is not considered here.

We do however want to include additional constraints that are imposed by the platform owner. Since they want to promote the growth of their platform, the churn rate of shippers should be minimized. In other words, they want to keep their customers happy, so that they continue to use the platform, thereby improving growth. The company has data available on the likeliness of shippers leaving the platform, and on the quality of service of the carriers. The inclusion of this data in the routing algorithm should increase the customer satisfaction.

Furthermore, the type of requests that are considered are requests that need to be picked up today and delivered tomorrow. This type is called AB requests. One important notice is
that when a request is assigned to a certain carrier, its delivery location is not considered. Therefore, it may occur that a request delivery may be difficult to fulfill the next day.

1.4. Research questions
The main research question that is formulated to investigate the problem mentioned in the motivation is:

RQ: What is the effect of varying collaboration levels in a dynamic vehicle routing framework with multiple objectives?

In order to answer the main research question, several subquestions have also been formulated, as listed below. These subquestions will aid in providing complete view of the research problem.

SQ 1: Which methods are available to model and solve the dynamic and collaborative vehicle routing problem?

SQ 2: How can the performance of the carrier routing be evaluated?

SQ 3: How can the level of horizontal collaboration be controlled?

SQ 4: How does the dynamic nature of the problem influence the routing performance?

SQ 5: At what cost can the additional objective of customer satisfaction be fulfilled?

1.5. Report structure
The work is structured as follows. Chapter 2 contains the literature study, in order to give a overview of the research field with respect to the vehicle routing problem and its variants. Chapter 3 states the problem formulation that represents the identified problem. Furthermore, the solution method is presented in chapter 4. Chapter 5 presents the experimental evaluation of the proposed solution technique. Its performance is monitored and the results are presented. The report is finalized by presenting the conclusion, discussions and future research in chapter 6.
Two problems were identified in the introductory chapter, namely the insertion of dynamic requests, and the routing optimization for horizontal collaboration. This chapter introduces the literature related to the vehicle routing problem. Firstly, by presenting a general introduction to the vehicle routing problem, and the different variants. Following that, three important aspects of the problem are research more in depth, being the dynamic, collaborative, and customer satisfaction aspects. Finally, multiple solution methods are presented.

2.1. Problem introduction
The type of problem considered in this work is part of a general class of problems called the vehicle routing problem (VRP). Introduced in 1959 as the truck dispatching problem [9], the VRP is an optimization problem in which a number of requests needs to be serviced by a certain number of vehicles at minimum cost. Examples of real-life cases are numerous, such as a mailman delivering letters and parcels, a mechanic servicing customers throughout a city, or a school bus bringing children to school. Depending on the problem, different boundary conditions are in place. The mailman, for example, prefers the shortest route through all the streets, the mechanic is bounded by the time windows of the appointments he has, and the school bus can only pick up as many children as there are seats in the bus.

Of course, these are not the only possible variations of the VRP. The possible extensions and variations of the general VRP are numerous [50]. In the problems we are facing the following attributes will be in place:

**Dynamic:** In the case of a dynamic VRP, the final number of requests is not known beforehand. During the day additional requests appear in the set of requests.

**Horizontal collaboration:** Horizontal collaboration is a VRP variation, where the separate VRPs of single carriers are investigated jointly, in order to provide additional gain by allowing exchange of requests between them.

**Customer Satisfaction:** The customers using the platform, should continue to do so, therefore they need to stay satisfied.

**Capacity:** The trucks used by the carriers are limiting the cargo size and weight that can be accumulated during a single route. Capacity constraints will ensure feasibility in terms of cargo dimensions and weight. The cargo dimension is defined in loading meters, a measure often used for palletized transport.

**Time windows:** The requests will have a time window assigned to it. In order to comply with the provided time windows, they need to be considered in both the insertion and the optimization.

**Multi depot:** The different carriers do not operate from the same depot. In a multi-depot problem, each carrier has his own.
2.2. Dynamic vehicle routing problem

A vehicle routing problem is considered static when all the input data of the problem are known before the routes are constructed. When the problem information changes over time, it is considered a dynamic vehicle routing problem (DVRP). Examples of dynamic properties are listed below [37]:

**Number of requests**: Most often encountered in dynamic VRPs is a changing number of requests. As is the case in this work, the number of requests increases over time.

**Travel times**: When arc traversing costs are based (partly) on travel times, instead of distance, dynamic travel times can be included. In this case the arc traversing costs $c_{ij}$ are varying over time. By including variable arc traversing costs, the effect of traffic jams, road accidents, or variable maximum speeds can be included realistically.

**Vehicle availability**: This type of dynamism can be included when the effect of vehicle breakdowns on the vehicle routing is investigated.

**Request demand**: More recently, dynamic request demand is a property where the amount of goods to be picked up or delivered is revealed dynamically.

In this work, we will focus solely on a dynamic number of requests, and the effect that it has on the solution methods. An exemplary figure of a DVRP with a dynamic number of requests is shown in figure 2.1.

### 2.2.1. Degree of dynamism

In order to classify dynamic VRPs, often the degree of dynamism (DOD) is used, as shown in equation (2.1). The DOD is the number of dynamic requests as a fraction of the total number of requests [33].

$$\delta = \frac{n_d}{n_{tot}} \tag{2.1}$$

The DOD can further be extended by taking into account how far ahead requests become known. This effective degree of dynamism (eDOD) is shown in equation (2.2), where $t_i$ and $T$ are the time at which request $i$ becomes known, and the total event horizon, respectively. DVRP problems are classified as weakly, moderately, and strongly dynamic for values of $\delta^e < 0.3$, $0.3 \leq \delta^e < 0.8$, and $\delta^e \geq 0.8$, respectively [29].

$$\delta^e = \frac{1}{n_{tot}} \sum_{i \in R} t_i \frac{1}{T} \tag{2.2}$$

In the case that time windows are present, the eDOD, can be used to reflect the level of urgency for a request. The reaction time is the difference between the time of disclosure $t_i$, and the end of the time window $l_i$, see equation (2.3).
2.2. Dynamic vehicle routing problem

\[ \delta_{tw}^e = \frac{1}{n_{tot}} \sum_{i \in R} \left( 1 - \frac{l_i - t_i}{T} \right) \]  

(2.3)

2.2.2. Consequences for solution methods

First of all, dynamically revealing information increases routing costs, compared to knowing all information beforehand. The loss of performance of a dynamic algorithm, compared to an ideal static algorithm which has all requests known beforehand, is expressed in the competitive ratio \( c_r [30] \), as shown in equation (2.4).

\[ c_r = \frac{c(A, I)}{c^*(I)} \]  

(2.4)

Where \( c(A, I) \) and \( c^*(I) \) are the cost found by the dynamic algorithm \( A \) for instance \( I \), and the optimal costs for instance \( I \) as found by an exact algorithm with full information. This quantification of loss-of-efficiency is useful to compare different routing algorithms.

However, lower bounds are generally difficult to obtain, due to the problem complexity. As will be explained in section 2.5, for a large number of requests it will take very long for an exact algorithm to find a solution. In order to still quantify the effect of dynamism on the problem, in the case that optimal solutions are not available, algorithms can be compared using a comparative analysis [27], as shown in equation (2.5).

\[ V_A = \frac{c(A, I)}{c^{off}(A, I)} \]  

(2.5)

Where \( c^{off}(A, I) \) is the cost found by algorithm \( A \) when provided with full information of instance \( I \).

Next to the competitiveness of the solution methods, the methods themselves are important to look into. The methods for solving DVRPs are often not different from the methods as used for static methods. Two approaches are identified for dealing with dynamic requests, periodic or continuous reoptimization [37].

Periodic reoptimization

In the case of periodic optimization, the solution algorithm produces a set of initial routes with the requests that are known at the start of the day. Then, at discrete time intervals, called decision epochs, a static VRP is solved. The decision epochs are either fixed periods, or whenever a new request appears. The advantage is that static methods can be applied to solve the current state of the problem. With extensive research done for the static VRP, numerous options are available. Furthermore, computational capacity is only used when needed, therefore minimizing the waste of computational resources. The main disadvantage is that no routing plan is available before the optimization has completed, thereby inducing additional delays for the carriers, possibly.

Exemplary is the work of [38], where a VRP variation, called the dial-a-ride-problem (DARP) is solved using an exact method. The works shows that reoptimization of the solution is favourable over insertion of new requests. Furthermore, it shows that for larger problem sizes computational times grow exponentially.

Continuous reoptimization

In contrary to periodic optimization, continuous optimization repeatedly tries to improve the current solution. A continuous approach keeps an adaptive memory of good performing solutions, which are updated when a new request appears. Compared to periodic optimization the advantages are maximization of computational capacity, which may allow a more complex solution method, and the real time availability of a feasible solution. The downside is that continuous approaches can only use heuristic approaches (section 2.5), since they rely on updating previous solutions.
2.3. Collaboration

Collaboration in general terms is when two or more entities share or exchange resources with each other, in order to realize benefits that they cannot realize individually [1]. Increasing the level of interaction between companies generally results in higher collaborative gains, but also requires more trust of both collaborators. Many different frameworks for describing varying levels of collaboration have been proposed, with varying steps between no collaboration up to complete integration between companies [28].

Often, two types of collaboration are distinguished, being vertical and horizontal collaboration, see figure 2.2. In the case of vertical collaboration, companies belong to different levels of the same supply chain. Often mentioned in the case of vertical collaborations, is the reduction of the bullwhip effect [12]. Interaction between customers and suppliers is used to reduce fluctuations in order forecasts along the supply chain.

Horizontal collaboration on the other hand, is when companies operating in the same layer of the supply chain join forces. Again as example, group purchasing of goods in order to have a stronger position in price negotiations. In this work we will only focus on horizontal collaboration, since we are dealing with actors all working on the same level of the supply chain.

2.3.1. Collaboration in road based transport

Compared to the maritime and aviation shipping, collaboration in road based transport is limited [8]. In maritime shipping for example, conferences are shipping lines that can be operated by all carriers that are in the conference. They operate the shipping line with equal rates and levels of service. These conferences discourage price wars and offer scale benefit due to the larger volume. Also in aviation collaborations are a common factor. The three major airline alliances, Star Alliance, SkyTeam and Oneworld, connect nearly all major carriers.

The higher occurrence of collaborations in these industries compared to road based transport can be explained by several factors. Firstly, the number of players in these industries is significantly lower, resulting in easier consolidation. Secondly, the assets used are more expensive (which also explains the lower number of players), requiring larger returns for maintaining the same return of investment. Finally, the average length of haulage is longer.

Since collaboration does not occur that often in road based transport, one can question what the added value of collaboration in that specific case is. Fortunately, numerous works
have researched the topic of collaborative gain. For several problem variants the added value of horizontal collaboration varies, depending on several factors.

Depending on the amount of information that carriers are willing to share with the planning authority, two approaches exist: centralized and decentralized planning [14].

2.3.2. Centralized planner

In centralized planning, the planner receives full information of all carriers involved, and aims at minimizing the total costs of the complete collaboration. The effect of having full information is that the central planner needs to solve a normal vehicle routing problem. The observed collaborative gain is largely dependent on the test instances.

Around 20-30% collaborative gain was found in the work of [7] which is most often found in other works as well. The centralized approaches of [52] and [51] however, found only gains around 10%. Next to economical gain, increased performance can also be found in terms of pollution, or congestion [42].

In cases where the initial requests are tightly clustered around a carrier’s depot, the added value of collaboration is lower than in cases where requests are distributed more evenly. It becomes apparent from figure 2.3, that requests that are easy to serve for multiple carriers, are favourable in the case of collaboration. On the contrary, would there be no collaboration, these requests are a likely cause for competition. The distribution of requests is therefore a measure for the level of competition, which leads to the conclusion that collaboration is more beneficial in highly competitive cases.

Next to the influence of geographical distribution, another significant parameter is the number of vehicles per carrier. The extreme cases of a single vehicle case and a unlimited vehicle case was investigated by [15]. This works shows that collaborative gain was 24.5% and 16.8% on average, for the single and multi vehicle case, respectively. This shows that the flexibility of having multiple vehicle decreases the collaborative gain.

2.3.3. Decentralized planner

In the case that collaborators are not willing to share full information, decentralized approaches provide solutions. In this case, the collaborators themselves decide which orders they want to trade with the planning authority. Thereby, they do not need to share information of requests that they do not want to trade. The actual exchange of the requests may be fulfilled either directly between carriers, or via an auction mechanism.

When comparing centralized with decentralized approaches, it is discovered that centralized approaches perform better than decentralized approaches. In test instances evaluated by [2], a decentralized approach performed 13-27% worse than a centralized approach, depending on the geographical distribution of customers. In the work of [51], performance loss between 4-19% was found.

The advantage of a decentralized approach over a centralized planner is evident. It ac-
commodates carriers that are anxious to join collaborations, by not forcing them to disclose complete information.

### 2.3.4. Collaboration fairness

The main argument that prevents carriers from joining collaboration initiatives, is the uneven request distribution that arises in collaborations. Together with a fear of losing customers and market share, these arguments hamper the real life application of collaboration. A fair distribution of requests is therefore preferred. By balancing the workload between carriers, the request distribution can be made fair.

Initially, workload balancing was investigated for single carrier routing problems, since non-monetary effects are obtained by doing so. These effects are; increased driver satisfaction, customer service and more routing flexibility [35]. When applied in a collaborative environment, these non monetary effects are still obtained, alongside the increased likeliness of collaborating.

Balancing is achieved by including additional constraints, or by penalization of the objective function. Exemplary balancing constraints are: equal tour length, equal tour duration, or equal loading rate. In the case of collaboration, useful balancing constraints are equal number of requests per carrier, equal number of initial requests per carrier, or equal profit per carrier.

### 2.4. Service levels

This section will briefly introduce the inclusion of customer satisfaction, in literature better known as service levels. The VRP with service levels is tightly related to the VRP with profits (VRPP), which is therefore introduced first.

The inclusion of service levels is mentioned since the platform owner wants to grow its market share, which is partially achieved by keeping shippers satisfied. This work, where multiple carriers are included, with each of them having a fixed level of service, is not previously encountered in the literature.

#### 2.4.1. Vehicle routing problem with profits

In the VRPP, service of a request is awarded a certain amount of revenue, at the expense of additional routing costs for serving that request. The objective of the VRPP, is to maximize the profit. However, depending on the specific problem, servicing all requests may not deem profitable. Servicing all requests may even be impossible in the case that additional constraints, such as capacity or time window constraints, are in place. In order to increase the amount of serviceable requests, several options exist. Often encountered is the outsourcing of unserviced requests, where requests are serviced by a third party [25]. Next to outsourcing, relaxation of time window constraints is another viable option.

#### 2.4.2. Time window relaxation

In most works found in literature, the service level is equivalent to how well the time window is met [47]. Deviation from the time window set by the customer, results in a lower level of service. An example of a soft time window is shown in figure 2.4. By relaxing the time windows specified by customers, the number of feasible solutions for the VRP increases. This generally results in lower routing costs. Compared to the VRPTW, where time windows are hard constraints, routing flexibility is larger for the VRPP with soft time windows.

### 2.5. Solution methods

This section first discusses the class of computational complexity that the VRP and its variations belong to. Additionally, it discusses both exact methods and heuristic methods. Although the computational burden for exact methods are higher, they guarantee to find the optimal solution. For heuristic methods, they are not guaranteed to find the global optimum, but do find good approximate solutions in reasonable time.
2.6. Exact methods

Although the computational complexity of the VRP is 'high', for small problems, exact methods can still be applied. For problems with a larger number of nodes, generally the computational time grows exponentially.

For solving the VRP and its variations to a global optimum, the exact solution methods fall into two classes, with the difference being the problem formulation. The problem is either formulated in its classical form, or as a set partitioning problem. Note that the VRP, depending on the attributes, generally is a mixed integer programming (MIP) problem, consisting of both integer and continuous variables. The integer decision variables include the binary choice if an arc is traversed, and the number of trucks used. Continuous variables may include the arrival and departure times.

Most exact methods for solving the VRP focus on branch and bound methods [49]. Notable is the branch-cut-price method found in [13], which for the first time was able to solve problems with up 100 customers. Its improved and extended versions are currently able to solve up to 350 customers [36]. Noted however, is that computational times are in the range
of multiple days, so it is impractical to apply to problems that require a solution quickly.

2.7. Heuristics
Although exact methods are able to find optimal solutions for small size problems, they do not scale very well. Therefore, heuristics are a viable alternative to find good approximations of the optimum. The concept of heuristic methods is simple. They make a perturbation of the current solution in order to see if it improves the solution. If the solution is improved, it is accepted as the new current solution and a new perturbation is made. In this section we will explore two groups of heuristics, local search and large neighborhood search.

2.7.1. Local search
In a local search only a small perturbation of the current solution is allowed. Because the number of possible perturbations is limited for local search operators, it is possible to explore all perturbations, and consequently the best is chosen as the new current solution.

Exemplary local search operators are listed below, and also shown in figure 2.6. This list is not exhaustive and further reference is made to [3]. Distinction is made between intra-route operators and inter-route operators. For intra-route operators only a single route of the VRP is modified, equivalent to modification of a Traveling Salesman Problem (TSP) problem. For inter-route operators nodes or arcs are exchanged between routes.

**2-opt** Using this operator, two arcs of the same route are removed, and reconnected in such a way that a new route is created. The consequence is that the reconnected part of the route is traversed in reverse.

**Or-opt** This operator cuts three arcs of the same route, and connects them in such a way that none of the old arc segments are traversed in reverse.

**relocate-opt** is an operator that picks a node from a route and inserts it in another route.

**2-opt** works the same as the normal 2-opt, but it cuts two arcs of two different routes, as compared to two arcs in the same route for 2-opt.

**exchange-opt** modifies the current solution by exchanging two nodes between two routes.

**cross-opt** is comparable to exchange-opt, however, it exchanges two arcs between to routes, instead of two nodes.

2.7.2. Large neighborhood search
As compared to local search, large neighborhood search (LNS) explores a bigger neighborhood of the current solution. Two effects are identified. Firstly, because the neighborhood is larger, the exploration of the neighborhood takes more time and it is therefore not possible to explore the neighborhood completely. Secondly, the tendency to get stuck in a local minimum is smaller.

Introduced by [45], LNS works by applying a destroy and repair method. The general idea is that part of the solution is ‘destroyed’, meaning that multiple nodes are removed from the solution. These removed nodes are subsequently re-inserted with the ‘repair’ method. The power of a LNS heuristic was shown by [44], where it was named as the ruin-and-recreate method. They were able to provide new best solutions to Solomon’s benchmark problems [46] in multiple cases.

Further improvement to LNS methods was achieved by [41], which they coined adaptive large neighborhood search (ALNS). Where previously only a single destroy and repair method were in place, or in the case of multiple, were selected randomly, their developed method incorporated a mechanism to select well performing operators more often. By making the operator selection method adapt to the specific problem, best known solutions to Solomons’s benchmark problems were found in 50% of the cases. ALNS has proven to be highly popular recently in the research field. Both due to its performance, and its adaptability to different problem variations [32] [5] [17].
2.8. Metaheuristics

One often encountered problem of heuristic methods, is that they converge to a local minimum. Since heuristic methods only accept improving solutions, these methods get stuck in local minima. In order to escape local minima, metaheuristic methods are often implemented on top of heuristics. They consist of a framework in which problem specific heuristic rules are implemented. Below listed are three of the most often implemented metaheuristic methods for the VRP [16].

2.8.1. Simulated annealing

Simulated annealing (SA) has its origin in the controlled cooling of metals, in order to obtain preferable mechanical properties [26]. Annealing is a process in which the temperature gradient during the cooling process determines the final mechanical properties of a metal.

Analogue to this annealing process, SA as optimization technique is able to search differently sized search spaces, depending on its settings. From an initial solution, a local search is performed. If the objective function improves, the solution is accepted directly. However, in order to not converge to local minimum, there is also a probability that a worse solution is accepted as solution. The probability of acceptance is dependent on the difference in objective function value between the two solutions and the 'temperature'. This 'temperature' is commonly a linear descending function. The result is that SA in the early stages of the optimization is able to search a large area of the search space, with the search space shrinking as the 'temperature' decreases as well.

2.8.2. Tabu search

Just like SA, tabu search is a local search based metaheuristic, being first investigated by [18]. Compared to local search, where only improving solutions are accepted, tabu search is able to accept worsening solutions. This is achieved by keeping a tabu list. The tabu list is a short term memory that keeps track of solutions that are previously visited. Tabu search is not allowed to move back to solutions that are already on the tabu list.

Tabu search works as following: firstly, the search space around the current solution is evaluated using the selected local search method; if the objective function improves, the new solution is accepted directly. However, when the current solution has converged into a local minimum, an improving solution cannot be found.

At this point a solution is accepted that is not improving. Since the algorithm is not allowed to move back to a solution that is on the tabu list, the algorithm moves away from...
the local minimum. The tabu list also makes sure that local cycling between local minima is evaded.

It is important to note that the tabu list length is finite. This is to prevent the algorithm from using excessive memory. Furthermore, it prevents the algorithm from entering very unpromising neighborhoods of the solution space.

2.8.3. Genetic algorithm

Inspired by the natural evolution of species, well known from Darwin’s theory, genetic algorithm (GA) is a solution method that applies this principle for difficult combinatorial problems. The application of GA for combinatorial problems, of which the VRP is one, was popularized in the work of [20].

In nature, the DNA encodes the characteristics that an individual has in a string of base pairs. GA also uses such a string to encode its population. Since the string representation can be adapted for virtually any kind of combinatorial problem, GA is widely applicable.

When VRP solutions can be encoded in strings, GA can imitate nature in order to evolve the population. This involves the following three steps:

**Initialization:** During the initialization, a large initial population of solutions is (randomly) generated. The size of the initial population should large enough so that it can explore the solution space completely, but not too large, since that would harm the algorithms efficiency.

**Selection:** The selection phase rates the population with some form of fitness function. In the case of VRP, these are generally the routing costs. According to the fitness function, only the fittest individuals of the population can survive. The rest of the population is discarded.

**Reproduction:** The reproduction phase generates the population’s new members. Reproduction can be implemented as cross over, mutation, or both. Cross over reproduction selects two parents from the population, and generates offspring by combining parts of both parents’ strings. Mutation is a random mutation of an individual. For both mutations and cross over there are numerous possible approaches.

The selection and reproduction steps are repeated until a termination criterion is met. An example of a GA implementation for a DVRP, can be found in [19].

2.8.4. Other

The previously mentioned metaheuristics are the most often applied and well known types. However there are numerous other metaheuristics that are not explored here. Furthermore, hybrid metaheuristics, that combine the strengths of multiple metaheuristics are also emerging. Next to hybrids combining several features from two metaheuristics, also hybridisation between exact and metaheuristic methods is possible.

2.9. Concluding remarks

The problem described in the introductory chapter is related to the well known VRP, for which relevant literature was presented in this chapter. The model variations and solution techniques found in the literature are numerous, but always the objective of the optimization problem is to minimize costs. In the literature study, special attention is given to the dynamic VRP, horizontal collaboration, and customer satisfaction. When placing this work in the research field, it is noted that this work will be the first to investigate the effect of varying levels of horizontal collaboration in a dynamic vehicle routing setting. Furthermore, customer satisfaction is included in a multi objective setting to address the practical applicability for the platform owner.

It was noted that in a dynamical VRP, heuristics are most often used. This is because heuristics can provide good solutions is reasonable time. Furthermore, the literature research has shown that the ALNS method is one of the most popular heuristic methods cur-
rently used in the research field. This is due to its good performance and easy adaptability to different VRP variations. Therefore, it is decided to be used in this work as well.
In this chapter the problem is formulated as used for the remainder of this work. This will be stated in the mathematical form of an optimization problem. Furthermore, differentiation between the static and dynamic problem formulations is made. Next to that, the collaborative and customer satisfaction objectives are formulated.

3.1. Problem formulation

This section presents the problem formulation, which is twofold. Firstly, the static problem is presented in its mathematical form, and subsequently its dynamic counterpart.

3.1.1. Static problem

Let $G = (V,A)$ be a directed graph, with $A$ as a set of arcs, and $V = R \cup D \cup L$ as a set of nodes, $R$ being the initial requests, $D$ being the dynamic requests, and $L$ being the depot nodes. Furthermore, set $R$ is divided in subsets $R_l \forall l \in \{1...L\}$, such that each depot has its own set of initial requests. The considered problem is not a pick-up and delivery problem (PDP), and therefore the requests are only pick ups.

Each request $i$ has an associated load $q_i$, time window $(e_i,l_i)$, in which service of the respective node should start, and service time $s_i$, representing the amount of time a vehicle needs to stay at the respective node to fulfill the request.

Let $K_l$ be a set containing all vehicles at depot $l$, and $K$ the set containing all subsets $K_l \forall l \in \{1...L\}$. Each vehicle has an associated capacity $Q$, which is the same for each vehicle. The vehicle availability is implicitly defined by the time window associated with the vehicle’s depot.

The mathematical formulation of the considered static problem is shown in equations (3.1). The objective of the optimization problem is to minimize the objective function, equation (3.1a), while serving all nodes.

Constraints (3.1b) ensure each node is visited only once. Constraints (3.1c) ensure, that the vehicle entering the node is the same as the one leaving it. Constraints (3.1d) make sure that a vehicle leaving a depot returns at the same depot.

Constraints (3.1e) ensure time continuity, and constraints (3.1f) ensure no violation of time windows.

Constraints (3.1g) and (3.1h) ensure load continuity and no violation of the capacity constraint, respectively.
Table 3.1: Overview of symbols used in problem formulation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>Directed graph $(V,A)$</td>
</tr>
<tr>
<td>$V$</td>
<td>Set of all nodes</td>
</tr>
<tr>
<td>$N$</td>
<td>Set of requests</td>
</tr>
<tr>
<td>$N_f$</td>
<td>Set of fixed requests</td>
</tr>
<tr>
<td>$N_{us}$</td>
<td>Set of requests that are not scheduled in solution $S$</td>
</tr>
<tr>
<td>$R$</td>
<td>Set of initial requests</td>
</tr>
<tr>
<td>$R_l$</td>
<td>Set of initial requests of depot $l$</td>
</tr>
<tr>
<td>$D$</td>
<td>Set of dynamic requests</td>
</tr>
<tr>
<td>$L$</td>
<td>Set of depots</td>
</tr>
<tr>
<td>$K$</td>
<td>Set of vehicles</td>
</tr>
<tr>
<td>$K_l$</td>
<td>Set of vehicles at depot $l$</td>
</tr>
<tr>
<td>$Q$</td>
<td>Vehicle capacity</td>
</tr>
<tr>
<td>$c_{i,j}^k$</td>
<td>Cost of travelling from node $i$ to node $j$</td>
</tr>
<tr>
<td>$t_{i,j}^k$</td>
<td>Duration of travelling from node $i$ to node $j$</td>
</tr>
<tr>
<td>$q_i$</td>
<td>Load of node $i$</td>
</tr>
<tr>
<td>$S_i^k$</td>
<td>Cumulative load of vehicle $k$ at node $i$</td>
</tr>
<tr>
<td>$B_i^k$</td>
<td>Start time of vehicle $k$ at node $i$</td>
</tr>
<tr>
<td>$e_i$</td>
<td>Earliest start time at node $i$</td>
</tr>
<tr>
<td>$l_i$</td>
<td>Latest start time at node $i$</td>
</tr>
<tr>
<td>$s_i$</td>
<td>Service time at node $i$</td>
</tr>
<tr>
<td>$x_{i,j}^k$</td>
<td>Decision variable of traversing arc $(i,j)$</td>
</tr>
<tr>
<td>$T$</td>
<td>Current time</td>
</tr>
<tr>
<td>$S$</td>
<td>Solution</td>
</tr>
</tbody>
</table>

Constraints (3.1i) eliminate subtours.

\[
\begin{align*}
\text{min} & \quad \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{i,j}^k x_{i,j}^k \\
\text{subject to} & \quad \sum_{k \in K} \sum_{i \in V} x_{i,j}^k = 1 \quad \forall j \in N \\
& \quad \sum_{i \in V} x_{i,k^h}^k - \sum_{i \in V} x_{i,h}^k = 0 \quad \forall h \in N, k \in K \\
& \quad \sum_{i \in V} x_{i,l_i}^k \leq \sum_{i \in V} x_{i,j}^k \leq 1 \quad \forall l \in L, k \in K_l \\
& \quad x_{i,j}^k (B_i^k + s_i + t_{i,j}) \leq B_j^k \quad \forall i \in V, j \in V, l \in L \\
& \quad e_i \leq B_i^k \leq l_i \quad \forall i \in V \\
& \quad S_i^k = \sum_{i \in V} x_{i,j}^k (S_j^k + q_i) \quad \forall j \in V, k \in K \\
& \quad S_i^k \leq Q^k \quad \forall i \in V, k \in K \\
& \quad \sum_{i \in S} \sum_{j \in S} \leq |S| - 1 \quad \forall k \in K, S \subseteq N, |S| \geq 2 \\
& \quad x_{i,j}^k \in \{0,1\} \quad \forall i, j \in V, k \in K
\end{align*}
\]
3.2. Horizontal collaboration

In order to investigate the effect of horizontal collaboration in the case of a dynamic VRP, the level of collaboration needs to be controlled. The level of collaboration is expressed in the number of initial requests that are reassigned between carriers. This can either be done

$$c_{ij}^k = \beta_1 d_{ij} + \beta_2 t_{ij} + \beta_3 p_{ij}^k + \beta_4 \max(0,(SS_i - CQ^k))$$  \hspace{1cm} (3.2)
via constraining, or penalization of the objective function. In this work we will do so by penalizing the objective function. This choice is made, because this flexibility may allow larger collaborative gains. Next to that, request reassignment that only leads to marginal gains are hereby discouraged.

It is noted that the penalty only occurs when a carrier serves a request that was previously assigned to another carrier. The dynamic requests do not have an initial carrier, and therefore do not cause the penalty to occur.

\[
p^K_i = \begin{cases} 
1 & \text{if } i \in R \setminus R_i, k \in K_i \\
0 & \text{otherwise}
\end{cases}
\] (3.4)

### 3.3. Customer satisfaction

Since the platform owner wants to grow its market share, it has an additional objective that shippers keep using the platform. One of the ways it wants to do this, is by providing the highest level of service as needed. Compared to the literature, where the level of service is determined by how well the time window is met, the inclusion of service level as presented below is novel, mainly because the platform owner has additional data that generally is not available.

The platform owner has data available of shippers’ inactivity risk. The inactivity risk is the probability that a shipper will not place another request in the following month. The inactivity risk is used as a single measure for the shipper satisfaction (SS). This is because it is currently the only measure available.

Next to the SS, the platform owner has measure for the carrier quality (CQ). The carriers cooperating with the platform have different levels of service quality. Reasons for lower service quality can be late for a pick-up or delivery, damaging cargo, or losing cargo. The carrier quality is determined by taking into account the number of operational errors caused by the carrier.

With both SS and CQ available, the growth of market share is guarded as follows. When a shipper has a high probability of churning, for its next shipment a high quality carrier is assigned, even though this would harm the routing efficiency. Of course for satisfied customers, it is not necessary to do so. This is reflected by adding the following part to the routing costs \[c^K_{ij},\] as also shown in equation (3.2):

\[
\max(0,(SS_i - CQ^K))
\] (3.5)

### 3.4. Including conflicting objectives

Sections 3.2 and 3.3 have investigated additional objectives for the VRP, next to the minimization of the routing costs which are classically considered. In order to deal with multiple objectives, two approaches can be used, as listed below:

- Firstly, one approach is that only a single objective is optimized. The other objectives are constrained within reasonable bounds. In the case of conflicting objectives, this generally results in the decision variables exploiting the constraint as much as possible, thereby minimizing the objective function.

- On the other hand, multiple objectives can be aggregated in a newly constructed objective function, consisting of a weighted sum of the respective objectives. Depending on the selected weights of the single objectives, the objective function value converges to a Pareto optimal solution. In a Pareto optimal solution a single objective can only be improved by increasing its relative weight in the objective function, but this will always be at the expense of the other objectives.

As already shown in equation 3.2, this work will use a weighted sum of objectives. The way this is represented is to express all objectives in terms of monetary value. This choice is made, because penalty values for collaboration and shipper satisfaction are easy to express in a monetary value. For example, as platform owner it is an interesting business question
to determine how much value is attached to the risk of losing a shipper. The weighing factors are built up as follows:

\( \beta_1 \): The factor \( \beta_1 \) should reflect the cost of traveling one meter. This factor reflects both fixed (insurance, taxes, depreciation) and variable costs (fuel, tires, maintenance). Only excluded are labor costs, which can directly be related to driving times.

\( \beta_2 \): Next to the material costs, another major expense is the labor cost. Since these are expressed as an hourly rate, \( \beta_2 \) is multiplied with the driving time between nodes. Generally it is assumed that the vehicle speed is constant, so that the driver’s hourly rate can be included in the cost of traveling one meter. However, depending on the driven route this may not always be true. Therefore these two factors are separated.

\( \beta_3 \): This factor is the penalty value for reassignment of initial requests between carriers. In the case that reassignment of initial requests between carriers is beneficial in terms of routing efficiency, the gain should be higher than the penalty value for it to actually occur. This is to prevent reassignments of requests with low marginal gains.

\( \beta_4 \): Lastly, the factor that will control the shipper satisfaction. It can be viewed as the importance of growing the platform market share at the expense of inefficient routing. Alternatively it can be viewed as the amount of risk, the platform owner is willing to take that a shipper will leave the platform.

### 3.5. Concluding remarks

This chapter presented the problem formulation that will be used for the remainder of this work, both in static and dynamic form. It is noted that solving the problem in its dynamic form actually consists of repeated evaluation of the static problem, with a growing number of requests per epoch.

Furthermore, the additional objectives of controlling the level of collaboration, and shipper satisfaction are presented. The complete objective function consists of a weighted sum of the following parts, as presented in equation (3.2):

- Driving distance
- Driving time
- Collaboration penalty
- Shipper satisfaction.
This chapter describes the solution method for the problem formulated in chapter 3. The general framework of the solution method is presented, next to the three main components, being the initial solution, the solution improvement by destroying and insertion, and the acceptance of the newly obtained solution. Next to the algorithm description, the chapter contains the validation of the algorithm, in order to assure a proper implementation, and its competitiveness. Lastly, two other solution methods are presented. These are used in chapter 5 for a comparative analysis.

4.1. General framework

The solution method is based on the Adaptive Large Neighborhood Search (ALNS), as proposed by [41]. The solution method needs to handle both the dynamic effect of additional requests, next to minimizing the number of reassignments. Since the platform owner has full information available, a centralized solution approach is applicable. In a centralized approach, the VRPs of all carriers are aggregated in a single, joint VRP.

The solution method consists of three steps. Firstly, an initial solution is generated once. Subsequently, the solution is optimized using the ALNS algorithm, in two steps, firstly partially destroying the solution and thereafter repairing it. Finally, the newly obtained solution
is evaluated with an acceptance criterion. A visual representation of the solution method is shown in figure 4.1.

4.2. Initial solution
Since we are working with a heuristic algorithm, an initial solution is required before the algorithm can start optimization. Initial solutions are made with construction heuristics, of which there are numerous. For example, one of the most often encountered construction heuristics is the Clarke-Wright construction heuristic [6].

In this work, a sequential cheapest insertion heuristic is used. This choice is made since it is also used during the optimization phase of the algorithm, see section 4.3.4. The choice of construction heuristic is a trade-off between speed and solution quality. On the one hand it is preferable to commence quickly with optimizing the solution, but a low quality initial solution needs more iterations in the optimization phase before reaching the same result as a high quality initial solution.

For the instances evaluated in this work it was noted that the initial solution quality did not deteriorate significantly to justify a slower construction heuristic. Therefore, sequential cheapest insertion is selected, because it is the fastest of the implemented repair heuristics. The initial solution is constructed by applying the sequential cheapest insertion heuristic on the set of initial requests $R$.

4.3. Adaptive Large Neighborhood Search
The optimization of the solution is done by the ALNS heuristic. ALNS is different from large neighborhood search [45], in that it does not implement a single destroy and a single repair heuristic, but implements multiple heuristics. Since multiple destroy and insertion heuristics are available, a selection method is needed. The selection is performed with a so called roulette wheel, as shown in section 4.3.1. An algorithmic overview of ALNS is provided in algorithm 1.

4.3.1. Roulette wheel selection
The roulette wheel is used for selecting a pair of destroy and insertion heuristics for the current iteration. The performance of a certain heuristic during previous iterations influences its probability of being selected, such that well performing heuristics are more often selected. The probability of selection for a certain heuristic is defined in equation 4.1.

$$p_j = \frac{\omega_j}{\sum_{i=1}^{k} \omega_i}$$ (4.1)

Where $p_j$ is the probability of selecting heuristic $j$, and $\omega_i$ are the weights of heuristics $i \in \{1, 2, ..., k\}$.

4.3.2. Adaptive weight adjustment
After solution acceptance, first it is checked if the solution has previously been obtained. If it was obtained previously, it is discarded directly, otherwise the performance of the destroy and repair heuristic are evaluated as follows. Depending on the solution quality, the heuristic’s score $\pi$ are adjusted with score weights $\sigma_1, \sigma_2, \sigma_3$, see Table 4.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$</td>
<td>The new solution is better than the current best solution</td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td>The new solution is better than the current solution</td>
</tr>
<tr>
<td>$\sigma_3$</td>
<td>The new solution is worse than the current solution, but was still accepted</td>
</tr>
</tbody>
</table>
Algorithm 1 Adaptive Large Neighborhood Search

1: procedure ALNS ($S, q_{min}, q_{max}, th$)  
2:   $i \leftarrow 0$  
3:   $S_{best} \leftarrow S$  
4:   while $i \leq l_{max}$ do  
5:     $S_{new} \leftarrow S$  
6:     $n_{remove} \leftarrow \text{remove_number}(q_{min}, q_{max})$  
7:       • Determine number of requests to remove  
8:       between interval $[q_{min}, q_{max}]$  
9:     $S_{new} \leftarrow \text{remove_heuristic}(S_{new}, n_{remove})$  
10:       • Select remove heuristic using roulette wheel,  
11:       and perform removal  
12:     $S_{new} \leftarrow \text{insert_heuristic}(S_{new})$  
13:       • Select insertion heuristic using roulette wheel, and  
14:       perform insertion  
15:     if accept($S_{new}, S_{best}, th$) then  
16:       if $S_{new} \not\in \{\text{solutions}\}$ then  
17:         $S_{new} \cup \{\text{solutions}\}$  
18:       if $c(S_{new}) \leq c(S_{best})$ then  
19:         $S_{best} \leftarrow S_{new}$  
20:     end if  
21:     adapt_score(remove_heuristic, insert_heuristic)  
22:       • Heuristic scoring is based on performance  
23:   end while  
24: end procedure
Initially, each heuristic has the same weight $\omega$ of being selected. As the optimization progresses, some heuristics are more successful in generating good solutions than others. This is reflected in their obtained score. After $i_{\text{update}}$ iterations, the weights of the heuristics are updated as follows:

$$
\omega_j = \omega_j (1 - r) + r \frac{\pi_j}{o_j}
$$

(4.2)

Where $\omega_j$ is the heuristic weight, $r$ is the reaction factor, $\pi_j$ is the heuristic’s score, and $o_j$ is the amount of times heuristic $j$ was selected in the last $i_{\text{update}}$ iterations. The reaction factor thereby controls the speed at which the weights adjust to their performance. For $r = 0$, no adaptation occurs, whereas for $r = 1$ the new weights completely depend on their performance for the last $i_{\text{update}}$ iterations.

4.3.3. Destroy heuristics

As mentioned, three destroy heuristics are used in the current implementation. The destroy heuristics partially destroy the solution by removing requests from the solution. The number of requests to be removed $n_{\text{remove}}$, varies per iteration, and is selected randomly on the interval $[q_{\text{min}}, q_{\text{max}}]$. The three implemented destroy heuristics are listed below.

**Random removal:** From the current solution, requests are removed randomly until the desired number of removed requests is reached. Generally, random removal performs poorly, but it is important for diversification of the search, which allows escaping local minima. The pseudocode for random removal is shown in algorithm 2.

**Algorithm 2 Random Removal**

1: function random_removal($S, N, n_{\text{remove}}$)
2: $W \leftarrow \emptyset$ \hspace{1cm} • Initialize empty set of requests to remove
3: while $|W| < n_{\text{remove}}$ do
4: \hspace{1cm} $i \leftarrow \text{random_request}(N \setminus (N_f \cup L \cup W))$ \hspace{1cm} • Only requests that are not fixed in $S$, or depot, or already to be removed are eligible
5: \hspace{1cm} $W \leftarrow W \cup \{i\}$
6: \hspace{1cm} end while
7: $S \leftarrow \text{remove_requests}(W, S)$
8: \hspace{1cm} return $S$
9: end function

**Worst removal:** Worst removal removes requests that are not well routed in the current solution, and works as follows. First, for all requests, the cost saving is calculated. The cost saving is calculated as the difference between the solution cost including and excluding the specific request, as shown in equation 4.3. To continue, all requests are sorted according to cost saving, and the requests with the highest cost savings are removed.

$$
W_i = c(S) - c_{-i}(S)
$$

(4.3)

Where $W_i$ is the sorting index, and $c(S)$ and $c_{-i}(S)$ are the solution costs including and excluding request $i$.

In order to overcome the removal of the same set of bad performing requests every time worst removal is used, some randomization is included. The chance of removal is dependent on the removal cost, so that requests with high removal cost have a higher chance of removal. The degree of randomization is controlled with the parameter $p_w$. Algorithm 3 shows the pseudocode.

**Related removal:** Related removal selects requests from the solution that are related to each other in some way. Related removal is initialized with a random request. For all other requests in the solution, the relatedness is determined, and the requests that are most related with the initial request are removed as well.
4.3. Adaptive Large Neighborhood Search

Algorithm 3 Worst Removal

1: function worst_removal($S, N, n_{remove}, p_w$)  
2:     $W \leftarrow \emptyset$  
3:     while $|W| < n_{remove}$ do  
4:         for $i \in N \setminus (N_f \cup L \cup W)$ do  
5:             $W \leftarrow remove_cost(i, S)$  
6:         end for  
7:         $y \leftarrow random_number(0, 1)$  
8:         $i \leftarrow W[\lfloor |W| \cdot y^{p_w} \rfloor]$  
9:         $W \leftarrow W \cup \{i\}$  
10:     end while  
11:     $S \leftarrow remove_requests(W, S)$  
12:     return $S$  
13: end function

The relatedness between requests depends on their location, their load quantity and their time windows. See equation 4.4 for a formal definition.

$$R_{ij} = \frac{d_{ij}}{M_d} + \frac{B_i - B_j}{M_t} + \frac{q_i - q_j}{M_q}$$ (4.4)

Where $a_1$, $a_2$, and $a_3$ are scaling factors, and $M_d$, $M_t$, and $M_q$ are the maximum values of the respective quantities for all requests in $N$.

As was the case with worst removal, some degree of randomization is included. Again, this is to prevent the same groups of requests to be removed over and over. The randomization of related removal is controlled separately, for which the parameter $p_r$ is used. See algorithm 4 for the pseudocode.

Algorithm 4 Related Removal

1: function Related_removal($S, N, p_r$)  
2:     $i \leftarrow random_request(N \setminus (N_f \cup L))$  
3:     $W \leftarrow \{i\}$  
4:     while $|W| < n_{remove}$ do  
5:         for $j \in N \setminus (N_f \cup L \cup W)$ do  
6:             $W \leftarrow relatedness(i, j)$  
7:         end for  
8:         $y \leftarrow random_number(0, 1)$  
9:         $i \leftarrow W[\lfloor |W| \cdot y^{p_r} \rfloor]$  
10:     end while  
11:     $S \leftarrow remove_request(W, S)$  
12:     return $S$  
13: end function

4.3.4. Insertion heuristics

The requests that are removed by the destroy heuristics are placed in the solution again by the insertion heuristics, of which there are five. If the previous solution was infeasible, the requests that were not in the current solution in the first place, are also inserted. The insertion heuristics are listed below.
Sequential cheapest insertion: From the unsorted list of requests to be inserted, the insertion cost of the first item is evaluated. The insertion cost is calculated as the difference between the solution cost by in and excluding the specific node in the solution at a certain location. The request is inserted at the location with the lowest insertion cost. Also see algorithm 5.

Algorithm 5 Sequential Cheapest Insertion

1: function sequential_cheapest_insertion($S, N_{us}$)
2:     for $i \in N_{us}$ do
3:         $P \leftarrow \text{insertion_positions}(i, S)$ \hspace{1em} \textbullet \hspace{0.5em} Find all possible insertion positions and associated costs for request $i$
4:         $P \leftarrow \text{sort_by_cost}(P)$
5:         $p \leftarrow \text{pick_first_element}(P)$ \hspace{1em} \textbullet \hspace{0.5em} Return cheapest insertion position for request $i$
6:         $S \leftarrow \text{insert}(p, S)$
7:     end for
8: end function

Parallel cheapest insertion: This operator is comparable to sequential cheapest insertion, however instead of evaluating the insertion costs for a single node and inserting directly, the insertion costs of all nodes are evaluated, and the node with the lowest insertion cost is selected and inserted. See algorithm 6.

Algorithm 6 Parallel Cheapest Insertion

1: function parallel_cheapest_insertion($S, N_{us}$)
2:     while $N_{us} \neq \emptyset$ do
3:         for $i \in N_{us}$ do
4:             $P \leftarrow P \cup \{\text{insertion_positions}(i, S)\}$ \hspace{1em} \textbullet \hspace{0.5em} Find all possible insertion positions and associated costs for all requests in $N_{us}$
5:         end for
6:         $P \leftarrow \text{sort_by_cost}(P)$
7:         $p \leftarrow \text{pick_first_element}(P)$ \hspace{1em} \textbullet \hspace{0.5em} Return cheapest insertion request and associated position
8:         $S \leftarrow \text{insert}(p, S)$
9:     end while
10: end function

K-regret insertion: The previously mentioned cheapest insertions focus on inserting requests that are currently cheapest to insert. The result is that insertion of expensive or difficult requests is postponed. This postponement may deteriorate the final solution when requests inserted last are the most difficult requests.

The regret insertion tries to negate this effect by comparing the best insertion location with the $K$ next-best insertion locations. The regret cost is calculated as the difference between the current best insertion costs and the $K$ next-best costs, as shown in equation 4.5.

$$K_i = \sum_{h=1}^{k} p_h - p_0 \quad (4.5)$$

Where $K_i$ is the regret value, $p_h$ is the cost of inserting at the $h$-th best insertion position, and $p_0$ is the cost of inserting at the best insertion position. The node with the highest regret value is subsequently inserted in the solution.
4.4. Solution acceptance

Implemented are 1-regret, 2-regret and full-regret. In the case of full regret, all requests in a vehicle route are used for calculation of the regret measure. Note that 0-regret is equivalent to parallel cheapest insertion. Reference is made to algorithm 7.

Algorithm 7 K-regret Insertion

1: function k_regret_insertion(S, Nus, k)
2:     while Nus ≠ ∅ do
3:         for i ∈ Nus do
4:             P ← P ∪ \{insertion_positions(i, S)\} • Find all possible insertion positions and associated costs for all requests in Nus
5:             P ← sort_by_cost(P)
6:             C ← k_regret_cost(P, k) • Calculate the regret based on the first k elements of P
7:         end for
8:     return S
9: end function

4.4. Solution acceptance

After insertion, it should be determined if the solution is accepted or not. Naturally, if the performance of the new solution is better than the current solution, it should be accepted. However, to prevent getting stuck in a local minimum, a more elaborate acceptance criterion should be in place.

Historically, simulated annealing (SA) is the most used acceptance criterion used for ALNS. However, recent research has shown that other acceptance criteria perform better than SA [43]. The best performing criteria from this research are Threshold Acceptance (TA) and Record to Record Travel (RRT). Especially when the number of iterations is limited [10], good solutions are obtained using RRT. Finding good solutions in limited time is extremely important because of the dynamic nature of the problem. Therefore it is decided to implement RRT as acceptance criterion.

Using RRT a solution is accepted if the gap between the current solution and the best known solution is smaller than the threshold \( th \). The threshold decreases linearly each iteration until a final value is obtained. RRT’s pseudocode is shown in algorithm 8.

Algorithm 8 Record to Record Travel

1: function RRT(Snew, Sbest, th)
2:     if (c(Snew) − c(Sbest)) < th \cdot c(Sbest) then
3:         return True
4:     else
5:         return False
6: end if
7: end function

4.5. Validation

In order to validate the current implementation of the algorithm, and to determine its performance, it has been tested with instances publicly available. The used test instances are the Solomon’s benchmark instances [46]. These instances are for a static capacitated VRP with time windows. Thereby, the dynamic effect is not considered. This is because there are no widely used test instances for a dynamic VRPTW available.

The benchmark consists of 56 instances with 100 requests per instance. The instances are grouped in 3 groups, R, C, and RC. For R-instances, the requests are random uniformly
distributed around a single depot. For C-instances they are clustered. For RC-instances they are semi-clustered, meaning that they consist of both randomly distributed requests and clusters. Further distinction per group is made by the index. For index 1, tight capacity and time window constraints are in place, allowing only a few requests being serviced per vehicle. For index 2, these constraints are not as tight, resulting in higher number of requests per vehicle.

The objective of the test instances is firstly to minimize the number of used vehicles, and subsequently to minimize the total distance. Since the algorithm is currently not capable of performing route minimization, it is added specifically for the algorithm validation. The algorithm is allowed a maximum of 3000 iterations.

4.5.1. Route minimization

For the validation of the algorithm, a route minimization step also needs to be implemented. Since route minimization is not needed for the actual experiment, only a simple route removal heuristic is implemented. Route minimization is done by randomly selecting a vehicle from the solution, and removing all requests served by that vehicle. The removed requests are subsequently inserted in the routes of the remaining vehicles during the next iteration of the optimization.

![Figure 4.2a](image1.png)  ![Figure 4.2b](image2.png)

(a) Relative number of routes  
(b) Relative distance

Figure 4.2: Validation results: 4.2a shows the relative number of routes with respect to the best known solution. 4.2b shows the relative distance with respect to the best known solution.

4.5.2. Results

The obtained results are shown in figures 4.2a and 4.2b. As can be seen in figure 4.2a, the results of the route minimization, are varying a lot between instances. Firstly, the clustered instances perform perfectly. However, in the R1 and RC1 average performance is not that good. The R2 and RC2 sets perform generally well with some outliers. Note that the total number of routes is low in these sets, a single additional route can already commit 33% extra. The average gap with the best known solutions is 7%.

For the results of the distance minimization, figure 4.2b shows that performance varies less between instance sets. In some cases, the distance is even less than found in the best known solution, This is the case when there are more routes included, compared to the best known solution. There are no new best known solutions found, The average gap with the best known solutions is 3%.

Considering that only 3000 iterations are performed, compared to the 25000 iterations used in [41], performance can be considered good. To know that good solutions are obtained with few iterations is favourable in the dynamic setting of the actual problem. Tables of all validation results can be found in appendix A.
4.6. Solution methods
The validation of the solution method already shows good performance. However, the problem considered during the validation is different from the one presented in chapter 3. The goal of the research is to investigate the influence of varying levels of collaboration in a dynamic VRP setting. In order to do so, a total of three solution methods are used as described below:

Destroy & Insert: The Destroy & Insertion (DI) method is the solution method as described in section 4. An initial solution is provided by solving the static VRP, and consequently dynamic requests are inserted. In between insertions, the solution method continuously optimizes the objective function.

The results obtained with the DI method are used to determine the collaborative gain in the dynamic VRP setting.

Simple insertion: Compared to the DI method, simple insertion (SI) is a more simple approach, where dynamic requests are placed in the solution using sequential greedy insertion. By not allowing removal of requests, optimization in between insertions is not allowed.

The biggest benefit of SI over DI is that carriers are never able to lose a dynamic request, since no optimization in between insertions is allowed. The benefit of not losing dynamic requests is of course at the expense of higher objective function. This method is preferable for carriers, because they are certain that they will never lose requests during the day, but can only gain more. At the expense of higher routing costs.

Full information: When using the full information method (FI), a single, static VRP is solved, in which all requests are known. In other words, an initial schedule is constructed using both initial and dynamic requests. This static case is not realistic, since it is impossible to know all request beforehand. However, this method will serve as a lower bound on the objective function for each test instance. The lower bound will vary with the level of collaboration.

The obtained results are used for a method comparison of both dynamic strategies next to a collaborative gain assessment (CGA). The CGA of the FI method checks if the collaborative gain in the static case is comparable to what is found in literature.

4.7. Concluding remarks
This chapter presented the problem formulation that will be used for the remainder of this work, both static and dynamic formulations. It is noted that solving the problem in its dynamic form actually consists of repeated evaluation of a static problem, with a growing number of requests per decision epoch.

Furthermore, the additional objectives of controlling the level of collaboration, and shipper satisfaction are presented. The complete objective function consists of a weighted sum of:

- Minimizing driving distance
- Minimizing driving time
- Minimizing the number of request reassignments
- Minimizing customer satisfaction

This chapter described the solution method used, which is based on the ALNS algorithm. It is modified to accommodate the problem variant encountered and a different acceptance criterion is used. The implemented destroy and repair operators are the same as found in [41]. The validation of the implemented algorithm has shown good performance, with an average gap of 3% w.r.t. the best known solutions in terms of routing distance.

Lastly, two other solution methods are presented that will be evaluated as well during the experimental evaluation, for a comparative analysis.
This chapter presents the results that are obtained by the implemented solution method on the instances constructed from Quicargo platform data. Firstly, the used input data and the implementation of the solution method are discussed. Furthermore, three experiments are conducted. Firstly, a comparison of the developed solution method with another dynamic method and a full information method. Secondly, a collaborative gain assessment for varying levels of collaboration in the dynamic setting. Lastly, the routing performance is evaluated while in- and excluding the customer satisfaction objective.

5.1. Input data
The following section presents the data as used for the experimental evaluation. A proper result starts with high quality input data. Real data is preferred over synthetic test data, since synthetic data may be constructed with a bias. There are three types of input data required, being the requests, the initial solution, and the vehicles. For the requests, the data is provided by Quicargo. The initial solution and the vehicle data are constructed artificially. This is because there are no carriers yet willing to cooperate, providing Quicargo with schedules and initial requests. Consequently, the number of carriers also has to be decided. The number of carriers considered in this work is three.

5.1.1. Requests
All requests used are received via the Quicargo platform. For the instances, a fraction of request data of four weeks is sampled. Firstly, a normal two week period in mid April. Secondly, a busy two week period around the start of July. These requests are used as both initial requests and dynamic requests.

Since there are no carriers willing to share their initial requests with Quicargo yet, these need to be constructed from requests received via the platform as well. The total set of requests for a specific test instance is determined as follows: From the first week all requests are used as initial requests. The second week’s requests are used as dynamic requests. The total set is obtained by summing the request sets of specific days from both weeks, such that Mondays are summed with Mondays and Tuesdays with Tuesdays and so on. See also table 5.1.

Each request has several attributes, as stated in table 5.2. The time_stamp attribute is not needed for initial requests. Therefore, the time_stamp for initial requests is set at 6:00, since each day starts at 6:00. Note also that two load attributes are in place, w and lm, which are used for load weight and dimensions. This is because a vehicle’s capacity may be violated either if the weight, or the dimensions of the cargo are too large. The time_serv is assumed to be equal for all requests and set at 15 minutes. Lastly, dynamic requests do not have an initial carrier, for these requests initial_carrier is set to None.

Furthermore, the geographical distribution [lat,lon] of requests is shown in figure 5.1. It only shows the request of the single instance F for clarity. Figures for other instances can
5. Experimental evaluation

Table 5.1: Overview of request data

<table>
<thead>
<tr>
<th>Instance</th>
<th>Number of requests</th>
<th>Date used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial</td>
<td>Dynamic</td>
</tr>
<tr>
<td>A</td>
<td>50</td>
<td>43</td>
</tr>
<tr>
<td>B</td>
<td>38</td>
<td>48</td>
</tr>
<tr>
<td>C</td>
<td>41</td>
<td>45</td>
</tr>
<tr>
<td>D</td>
<td>37</td>
<td>34</td>
</tr>
<tr>
<td>E</td>
<td>48</td>
<td>19</td>
</tr>
<tr>
<td>F</td>
<td>56</td>
<td>63</td>
</tr>
<tr>
<td>G</td>
<td>53</td>
<td>64</td>
</tr>
<tr>
<td>H</td>
<td>55</td>
<td>58</td>
</tr>
<tr>
<td>I</td>
<td>40</td>
<td>48</td>
</tr>
<tr>
<td>J</td>
<td>64</td>
<td>54</td>
</tr>
</tbody>
</table>

Table 5.2: Request attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique identifying number</td>
</tr>
<tr>
<td>time_stamp</td>
<td>Time at which the request is received.</td>
</tr>
<tr>
<td>lat</td>
<td>Latitude</td>
</tr>
<tr>
<td>lon</td>
<td>Longitude</td>
</tr>
<tr>
<td>w</td>
<td>Weight of request</td>
</tr>
<tr>
<td>lm</td>
<td>Loading meters of request</td>
</tr>
<tr>
<td>time_from</td>
<td>Start of time window</td>
</tr>
<tr>
<td>time_to</td>
<td>End of time window</td>
</tr>
<tr>
<td>time_serv</td>
<td>Service time of request</td>
</tr>
<tr>
<td>inactivity_risk</td>
<td>Inactivity risk of the shipper</td>
</tr>
<tr>
<td>depot</td>
<td>Boolean identifier for depot nodes</td>
</tr>
<tr>
<td>initial_carrier</td>
<td>Initial carrier of the request.</td>
</tr>
</tbody>
</table>

be found in appendix B.

Next to that, some statistical information of the requests is shown in figures 5.2 and 5.3. Figure 5.2 presents information for both initial and dynamic requests, and figure 5.3 additional information that is only applicable for dynamic requests.

Figure 5.2 shows that the bulk of requests have a time window length that corresponds with standard working hours, ranging between 8 to 10 hours. Time windows shorter than 8 hours do not occur that often, and longer than 10 hours even less. Next to that, figure 5.2 also shows that the bulk of requests have a loading rate of less than 10%. The loading rate is expressed as fraction of the vehicle capacity. Since vehicle capacity is both constrained by weight and loading meters, the maximum value of both is used in this figure. For example, if a specific request uses 5% of the vehicle weight capacity, but occupies 10% of its loading meters, it shows as 10% in the figure.

Figure 5.3 shows that peak hours for submitting requests are in the morning between 7 and 9, sharply dropping until 11. After 11, the submitting of requests increases a bit, with finally a gradual decrease until the end of the day at 17. Next to that, many of the dynamic requests have a time window that closes between one and two hours after receiving the
5.1. Input data

The effective degree of dynamism (see equation 2.3), is \( \delta_{\text{tw}} = 0.39 \) for all requests. Although this seems like a low number, the effective degree of dynamism of the dynamic requests is \( \delta_{\text{tw}} = 0.79 \). So, around 50% of the total number of requests are highly dynamic requests, and the other half of requests are initial requests with \( \delta_{\text{tw}} = 0 \).

5.1.2. Carriers: depot and initial schedule

The initial requests are assigned to a carrier, of which there are three. It is assumed that the carriers each work from a single depot. The carrier’s depot locations are selected from carriers that are currently working with the Quicargo platform. The three carriers are selected so that their depot locations are roughly in an equilateral triangle, with the depot locations shown in figure 5.1.

Next to the requests, an initial schedule per carrier needs to be available. Since there are no carriers yet that are willing to provide their schedule, the initial schedules need to be created artificially. It is assumed that the three carriers are equally sized, and therefore they have more or less the same amount of initial requests. The initial requests of a single day are randomly assigned to one of the carriers. Subsequently, the initial solution is made by applying the implemented solution method as presented in chapter 4 to the initial set of requests. This will result in an optimized schedule per carrier. However, it may not reflect reality properly, because different carriers may apply different scheduling strategies.
5.1.3. Vehicles

It is assumed that each carrier has enough vehicles to serve all requests. All the vehicles are equal in terms of capacity and speed, so that driving distances and times are equal among vehicles. Each vehicle has a weight capacity of 22000 kg and a loading meter capacity of 13.6 meter. An overview of the vehicle attributes is shown in Table 5.3.

In other words it is often assumed that vehicle speed is constant, which results in time and distance matrices being equal or a scaled multiple of each other. However, this is not a proper assumption, since the vehicle speed is dependent on a specific route. The shortest route is not always the fastest route. The routing server (section 5.2.1) takes into account both the maximum speed of the vehicles, and the maximum speed allowed on the roads, resulting in unique distance and time matrices.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle_id</td>
<td>Unique vehicle identifier</td>
</tr>
<tr>
<td>company_id</td>
<td>Identifier of vehicle owner</td>
</tr>
<tr>
<td>max_w</td>
<td>Weight capacity</td>
</tr>
<tr>
<td>max_lm</td>
<td>Loading meter capacity</td>
</tr>
<tr>
<td>carrier_quality</td>
<td>Carrier quality</td>
</tr>
</tbody>
</table>

5.2. Experimental setup

The experimental setup consists of the software implementation of the solution method and its parameters, the hardware used for the simulations and, the different simulation strategies to make a proper analysis of the research problem.

5.2.1. Software implementation

The solution method is implemented in the Python programming language, which was chosen for its ease of use. The pros are: easy to produce code and to interact with the routing software, with the use of external packages. Furthermore, since it is a general purpose programming language (compared to MATLAB), greater flexibility in terms of implementation was obtained.

One of the downsides is that Python is relatively slow when compared to other programming languages, especially compiled languages such as C++, which are often encountered in the research field. Another difficulty is that, the Python interpreter has a global interpreter lock (GIL), which prevents sharing of memory between computing processes. This compli-
cates the use of multiple threads. With a multi-thread implementation, the algorithm’s iterations can be parallelized, speeding up the algorithm, and thereby improving its performance. Because of the GIL, it was decided not to use multi-thread implementation. In order to still make use of multiple threads, multiple test instances run side by side.

The routing software is provided by OSRM, an open-source alternative to Google Maps. The routing software provides the driving time and distance matrices for the requests that are in a specific instance. The routing software runs locally, which results in near instant time and distance matrices. Since OSRM is developed only for Linux, a virtual machine is used as Linux server.

5.2.2. Hardware
The simulations are performed on an Intel Core i7-6700HQ processor, with 8 available threads, and 8 GB of RAM. Since one thread is used for the routing software, up to 7 instances can be evaluated simultaneously.

5.2.3. Algorithm parameters
In chapter 4 several algorithm parameters were introduced, for which values need to be determined. The performance of the solution approach depends for a great deal on these parameters. Since the ALNS implementation described in the previous chapter largely follows [41], the algorithm’s parameters are largely kept the same. An overview of relevant parameters is presented in table 5.4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega$</td>
<td>Initial weights of heuristic</td>
<td>[1]</td>
</tr>
<tr>
<td>$\sigma_1, \sigma_2, \sigma_3$</td>
<td>Score weights</td>
<td>[33, 9, 13]</td>
</tr>
<tr>
<td>$r$</td>
<td>Reaction factor</td>
<td>0.1</td>
</tr>
<tr>
<td>$[q_{min}, q_{max}]$</td>
<td>Bound for $n_{remove}$</td>
<td>$[4, 0.35 \cdot</td>
</tr>
<tr>
<td>$p_w$</td>
<td>Randomization penalty for worst removal</td>
<td>3</td>
</tr>
<tr>
<td>$p_r$</td>
<td>Randomization penalty for related removal</td>
<td>6</td>
</tr>
<tr>
<td>$\alpha_1, \alpha_2, \alpha_3$</td>
<td>Relatedness weights</td>
<td>[9, 3, 2]</td>
</tr>
<tr>
<td>$[th_{min}, th_{max}]$</td>
<td>Bounds for $th$</td>
<td>$[0, 0.03]$</td>
</tr>
<tr>
<td>$i_{max}$</td>
<td>Maximum number of iterations, (see below)</td>
<td>3000</td>
</tr>
<tr>
<td>$i_{update}$</td>
<td>Number of iterations before weight adjustment</td>
<td>100</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>Obj. function weight: distance [22] [21]</td>
<td>0.1</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>Obj. function weight: time [22] [21]</td>
<td>0.62</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>Obj. function weight: collaborative penalty, (see below)</td>
<td>$[0, 3600, 8350, 10000000]$</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>Obj. function weight: customer satisfaction penalty</td>
<td>$[0, 10000000]$</td>
</tr>
</tbody>
</table>

Maximum number of iterations
As stated in section 2.5, it is not guaranteed that heuristic methods reach the global optimum. However, in order to make a proper comparison of the different simulation methods, all of them should be as close to optimality as possible. This is especially true for the static method, since its initial schedule is constructed using both initial and dynamic requests. Therefore, the maximum number of iterations for optimizing the initial schedule should be determined.

The number of iterations will be determined by using the static method, since this method uses all requests for the initial schedule. For a higher number of requests, more iterations are needed in the optimization phase. As an initial estimate, 3000 iterations are used, the
same amount as used for the model validation (section 4.5). This number is chosen because the total instance size is approximately the same as that of the instances used for validation. Then, with every step, the number of iterations is doubled, until no significant improvement is obtained.

The result is presented in figure 5.4. For 6000 iterations no significant improvement can be made, compared with 3000 iterations. Consequently, for all strategies a total of 3000 iterations will be used. For the static method this will be with the total set of requests, and for both dynamic methods this will be with the set of initial requests only.

Simulation speed
The performance of the three methods is determined by evaluating the test instances. For the DI method, this needs to be done in real time. The depot is open 18 hours per day for all instances, thus a single evaluation takes 18 hours as well. Since each instance is evaluated multiple times, this would lead to enormous computational times. Therefore, an investigation is done on the influence of the simulation speed on the routing performance. By simulating faster than real time, the amount of time between insertions is reduced. Since a single iteration takes a constant amount of time, irrespective of simulation speed, the number of iterations performed between insertions is reduced. When the routing performance does not deteriorate significantly with increased simulation speed, it is assumed safe to increase the simulation speed.

To investigate this effect all 10 instances are evaluated at multiple simulation speeds. The results are presented in figure 5.5.
As can be seen in figure 5.5, there is no significant decrease in the routing costs at lower simulation speeds. The routing performance does not deteriorate with simulation speeds up to 128. Therefore the remainder of the simulations with the DI method are performed with a simulation speed of 128. The simulation speed was not increased further for two reasons. Firstly, an increase of 128 already keeps total simulation time manageable. Secondly, erratic behaviour of the OSRM server at higher speeds was causing the algorithm to crash.

Collaboration penalty

The objective function weight $\beta_3$ is the parameter that ultimately controls the number of requests that are reassigned, and thereby the level of collaboration. Therefore, in order to control the level of collaboration properly, a sensitivity analysis is performed. After the sensitivity analysis, four values for $\beta_3$ are chosen. Two that will reflect two intermediate levels of collaboration, and two for full- and no collaboration. These will be used in the remainder of this work.

For the level of collaboration the variable $\xi$ is introduced. It is used as a relative measure for the number of reassigned requests. Thus, in the case that full collaboration is allowed, $\xi = 1$, reassignment is not penalized. Would $\xi = 0$ be, no requests are reassigned, thereby not allowing collaboration. The goal of the sensitivity analysis therefore is to determine values for $\xi = 0.66$ and $\xi = 0.33$.

The sensitivity analysis is performed using the static optimization method with only the initial requests. The dynamic counterpart is not incorporated since the collaborative penalty can not be incurred by the dynamic requests. All instances are evaluated a single time, with $\beta_3$ increasing in steps of 1000, until no requests are rerouted in all instances. When no requests are reassigned for all instances, it is assumed that the corresponding penalty value signifies ‘no collaboration’.

The results are shown in figure 5.6. As can bee seen, for different instances, the obtained results vary greatly. First of all, the value of $\beta_3$ to prevent all collaboration of instance F is more than 2.5 times higher than for instance J. Furthermore, some instances, such as J, show a gradual decline, whereas others, such a A, show plateauing. Also, some instances do not show continuous descent, instead spiking up sometimes. Next to that, the fraction of requests that are reassigned at $\xi = 1$, ranges between 41% and 88%, for instances B and E, respectively. The fraction of reassigned requests seems unrelated to the number of initial requests.

By normalizing the results of figure 5.6, figure 5.7 is obtained. Logically, the same variance per instance as before is present. However, by averaging the result over all instances, see figure 5.7b, a first indicator is available for $\beta_3$ at $\xi = 0.66$, and $\xi = 0.33$, being $\beta_3 = 3600$ and $\beta_3 = 8350$, respectively. These values are obtained by interpolation of the average result.

Concluding, it is difficult to determine two weights $\beta_3$ for intermediate levels of collaboration that work well for all instances by using this approach. Therefore, a different approach
is also tried. It is done by determining which fraction of the total routing costs are made up by the collaborative penalty. These results are shown in figure 5.8.

Again, there is no clear indicator which objective function weight \( \beta \) corresponds with \( \xi = 0.66 \) and \( \xi = 0.33 \). The contribution of the collaborative penalty ranges between 1.2% and 14.9%, with the collaborative penalty making up the largest part of the objective function for penalty values ranging between \( \beta = 2000 \) and \( \beta = 15000 \). When reviewing the previously determined values of \( \beta = 3600 \) and \( \beta = 8350 \), figure 5.8b shows that the collaborative penalty makes up 10% of the routing costs on average, which is fairly constant on this interval.

Since the second approach is also not conclusive it is decided that further analysis is performed with penalty values 3600 and 8350, for collaborative levels of \( \xi = 0.66 \), and \( \xi = 0.33 \), respectively.

5.2.4. Small scale experiment

The small scale experiment is performed as an additional validation of the proposed method. As noted, the implemented heuristic methods are not guaranteed to find optimal solutions, so by including this small scale experiment an additional check on the competitive ratios is available.

For the small scale experiments only 9 initial requests are present, and 10 dynamic requests. The small scale instance is constructed by modifying instance A. The algorithm’s parameters are the same as used for the remainder of the experiments, as shown in table 5.4.
5.3. Results

The obtained results for the instances shown in table 5.1 are presented in this section. Each method is evaluated for four levels of collaboration. All 10 instances are evaluated 10 times, for a total of 1200 simulations. In the simulations, customer satisfaction is not taken into account, since this may obfuscate the results. Customer satisfaction will be considered in a separate investigation, in section 5.3.5 resulting in another 100 simulations. Section 5.2.3 showed that the routing performance does not deteriorate with a higher simulation speed, therefore all simulations are sped up 128 times. The routing performance is expressed in terms of the routing costs, i.e. the first two terms of the objective function:

$$\beta_1 d_{i,j} + \beta_2 t_{i,j}$$

The obtained results are used for separate investigations. First of all, a comparison of the different simulation methods is performed. This is to determine the competitiveness of both the SI method and the DI method compared with the FI method, see also section 2.2.2.

The same set of results is also used for a collaborative gain assessment (CGA) per method. The results using the FI method are compared with the literature, and both dynamic strategies are consequently compared to the full information method.

5.3.1. Method comparison: performance

Figures 5.10 and 5.11 show the method comparison for instances E and G. For these instances the competitive ratio is smallest and largest respectively, for the DI method. The figures for all other instances can be found in appendix C. A table of competitive ratios is

<table>
<thead>
<tr>
<th>Method</th>
<th>$\xi = 1$</th>
<th>$\xi = 0.66$</th>
<th>$\xi = 0.33$</th>
<th>$\xi = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>1.027</td>
<td>1.283</td>
<td>1.020</td>
<td>1.222</td>
</tr>
<tr>
<td></td>
<td>DI</td>
<td>SI</td>
<td>DI</td>
<td>SI</td>
</tr>
<tr>
<td></td>
<td>1.052</td>
<td>1.282</td>
<td>1.026</td>
<td>1.331</td>
</tr>
</tbody>
</table>

Discussion

The results of the small scale experiment are shown in figure 5.9 and table 5.5. Figure 5.9 shows that DI performs better than SI in all instances. Next to that, it shows that the performance of DI (+3.1% avg.) is much closer to unity than is the case for SI (+28.0% avg.). This shows that DI has a clear advantage over SI, and that there is not much more to gain in terms routing efficiency.
Table 5.6: Overview of performed experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>( \beta_3 )</th>
<th>( \beta_3 )</th>
<th>Methods used</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>FI, DI, SI</td>
<td>Full collaboration excluding CS</td>
</tr>
<tr>
<td>2</td>
<td>3600</td>
<td>0</td>
<td>FI, DI, SI</td>
<td>High collaboration excluding CS</td>
</tr>
<tr>
<td>3</td>
<td>8350</td>
<td>0</td>
<td>FI, DI, SI</td>
<td>Low collaboration excluding CS</td>
</tr>
<tr>
<td>4</td>
<td>10000000</td>
<td>0</td>
<td>FI, DI, SI</td>
<td>No collaboration excluding CS</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>10000000</td>
<td>DI</td>
<td>Full collaboration including CS</td>
</tr>
</tbody>
</table>

shown in table 5.7.

![Figure 5.10: Routing costs of three strategies for instance E at varying collaborative levels](image)

![Figure 5.11: Routing costs of three strategies for instance G at varying collaborative levels](image)

Discussion

The results presented in this subsection show that the DI method outperforms the SI method in all instances and for all collaborative levels. Furthermore, the competitive ratio is clearly instance dependent. Reviewing table 5.7 per column shows that the lowest and highest competitive ratios are always found for instance E and G, respectively. This can be explained as a result of the number of dynamic requests per instance. Instance E only has 19 dynamic requests, which is the lowest of all instances, and instance G has 64 dynamic requests, which is the highest of all instances.

Furthermore, by increasing the objective function weight \( \beta_3 \), the number of reassignments is penalized. This results in a smaller solution space. This decrease of routing flexibility is
Table 5.7: Competitive ratios for all instances at all collaborative levels. Ratios for both the destroy & insertion method (DI), and the simple insertion method (SI) are determined with respect to the full information (FI) method. Per column, the minimum and maximum are highlighted in bold. Note that highlighting is based on unrounded values.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\xi = 1$</th>
<th>$\xi = 0.66$</th>
<th>$\xi = 0.33$</th>
<th>$\xi = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.52</td>
<td>1.68</td>
<td>1.47</td>
<td>1.38</td>
</tr>
<tr>
<td>B</td>
<td>1.43</td>
<td>1.68</td>
<td>1.41</td>
<td>1.41</td>
</tr>
<tr>
<td>C</td>
<td>1.40</td>
<td>1.69</td>
<td>1.45</td>
<td>1.42</td>
</tr>
<tr>
<td>D</td>
<td>1.30</td>
<td>1.57</td>
<td>1.31</td>
<td>1.25</td>
</tr>
<tr>
<td>E</td>
<td>1.27</td>
<td>1.49</td>
<td>1.23</td>
<td>1.21</td>
</tr>
<tr>
<td>F</td>
<td>1.54</td>
<td>1.78</td>
<td>1.48</td>
<td>1.44</td>
</tr>
<tr>
<td>G</td>
<td>1.58</td>
<td>1.84</td>
<td>1.63</td>
<td>1.57</td>
</tr>
<tr>
<td>H</td>
<td>1.54</td>
<td>1.86</td>
<td>1.53</td>
<td>1.46</td>
</tr>
<tr>
<td>I</td>
<td>1.45</td>
<td>1.79</td>
<td>1.47</td>
<td>1.49</td>
</tr>
<tr>
<td>J</td>
<td>1.36</td>
<td>1.63</td>
<td>1.38</td>
<td>1.32</td>
</tr>
<tr>
<td>Average</td>
<td>1.44</td>
<td>1.70</td>
<td>1.44</td>
<td>1.40</td>
</tr>
</tbody>
</table>

Also apparent in Table 5.7. For the full collaborative case, competitive ratios are higher than is the case for the non-collaborative case.

5.3.2. Method comparison: computational times

Next to the performance of the different strategies in terms of routing costs, another important factor to investigate is the computational times of the different strategies. For the real time applicability of the different strategies, computational times should not be too excessive. Figure 5.12 shows the computational times for different strategies.

Discussion

First of all, it is clear that the SI method is the fastest in all instances. Then, the DI method is generally fastest, except instance D. Lastly, FI is slowest. This shows that inserting requests is a much faster operation than optimization. Next to that, it is evident that dealing with an increasing number of requests, which is the case in FI, increases the computational time per iteration significantly. When comparing FI, and DI, FI is slower still although it only solves a VRP a single time. DI on the other hand does an initial optimization with less requests, which is faster, but when adding the destroy and insertion, it is still faster, since it only iteratively
5. Experimental evaluation

optimizes a partially fixed solution. Next to the average computational times, figure 5.12 shows that the variance grows with instance size for the FI method.

To conclude, inserting a request happens near instantly, so that both dynamic methods always provide a feasible solution when a new request appears. Next to that, the time needed to provide an optimized solution using DI is also low enough such that it does not limit the solution quality. It is expected that for larger instances a dynamic method has better scalability than a static method.

5.3.3. Collaborative gain: routing cost

Next to analysing the performance of the aforementioned strategies, a collaborative gain assessment for different levels of collaboration is performed. Section 2.3 showed that collaborative gain in static cases generally ranges between 20-30% for full collaborative cases.

Table 5.8: Collaborative gain for each method; Full information (FI), destroy & insertion (DI), and simple insertion (SI) and three levels of collaboration. For each method, each level is compared with the non collaborative case $\xi = 1$. Per column, the minimum and maximum are highlighted in bold. Note that highlighting is based on unrounded values

<table>
<thead>
<tr>
<th>Method</th>
<th>FI $\xi = 1$</th>
<th>FI $\xi = 0.66$</th>
<th>FI $\xi = 0.33$</th>
<th>DI $\xi = 1$</th>
<th>DI $\xi = 0.66$</th>
<th>DI $\xi = 0.33$</th>
<th>SI $\xi = 1$</th>
<th>SI $\xi = 0.66$</th>
<th>SI $\xi = 0.33$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.65</td>
<td>0.68</td>
<td>0.75</td>
<td>0.75</td>
<td>0.76</td>
<td>0.79</td>
<td>0.78</td>
<td>0.81</td>
<td>0.85</td>
</tr>
<tr>
<td>B</td>
<td>0.73</td>
<td>0.75</td>
<td>0.78</td>
<td><strong>0.83</strong></td>
<td><strong>0.84</strong></td>
<td>0.88</td>
<td><strong>0.89</strong></td>
<td>0.90</td>
<td>0.89</td>
</tr>
<tr>
<td>C</td>
<td>0.70</td>
<td>0.71</td>
<td>0.77</td>
<td>0.76</td>
<td>0.80</td>
<td>0.85</td>
<td><strong>0.74</strong></td>
<td><strong>0.68</strong></td>
<td><strong>0.79</strong></td>
</tr>
<tr>
<td>D</td>
<td>0.68</td>
<td>0.68</td>
<td>0.76</td>
<td>0.75</td>
<td>0.76</td>
<td>0.80</td>
<td>0.82</td>
<td>0.77</td>
<td>0.80</td>
</tr>
<tr>
<td>E</td>
<td><strong>0.63</strong></td>
<td><strong>0.66</strong></td>
<td><strong>0.73</strong></td>
<td><strong>0.68</strong></td>
<td><strong>0.69</strong></td>
<td><strong>0.76</strong></td>
<td>0.77</td>
<td>0.77</td>
<td>0.81</td>
</tr>
<tr>
<td>F</td>
<td><strong>0.75</strong></td>
<td><strong>0.77</strong></td>
<td><strong>0.88</strong></td>
<td>0.82</td>
<td>0.81</td>
<td>0.91</td>
<td>0.85</td>
<td>0.86</td>
<td>0.91</td>
</tr>
<tr>
<td>G</td>
<td>0.71</td>
<td>0.72</td>
<td>0.82</td>
<td>0.81</td>
<td>0.84</td>
<td><strong>0.92</strong></td>
<td>0.84</td>
<td>0.88</td>
<td>0.92</td>
</tr>
<tr>
<td>H</td>
<td>0.68</td>
<td>0.70</td>
<td>0.80</td>
<td>0.80</td>
<td>0.81</td>
<td>0.89</td>
<td>0.84</td>
<td>0.84</td>
<td>0.86</td>
</tr>
<tr>
<td>I</td>
<td>0.69</td>
<td>0.70</td>
<td>0.77</td>
<td>0.74</td>
<td>0.76</td>
<td>0.85</td>
<td>0.86</td>
<td><strong>0.94</strong></td>
<td><strong>0.97</strong></td>
</tr>
<tr>
<td>J</td>
<td>0.65</td>
<td>0.67</td>
<td>0.81</td>
<td>0.73</td>
<td>0.77</td>
<td>0.89</td>
<td>0.77</td>
<td>0.80</td>
<td>0.90</td>
</tr>
</tbody>
</table>

Discussion
The collaborative gain per method and level of collaboration is presented in table 5.8. First of all, it is clear that decreasing $\xi$ generally results in less collaborative gain. Full collaboration
5.3. Results

Decreases routing costs between 18% and 31%. For $\xi = 0.66$, routing costs only increase slightly, ranging between 18% and 30%. Moreover, a high collaboration penalty ($\xi = 0.33$) decreases the collaborative gain severely for all strategies, with numbers ranging from 13% to 21%.

The averaged result of all instances is shown in figure 5.13. It shows that between full and high collaboration, routing costs only increase with 1%. Most collaborative gain is lost between low collaboration and no collaboration.

Lastly, the CGA shows no instance dependency. For none of the instances, does a single instance show maximum or minimum collaborative gain throughout the complete range of objective function weights $\beta_i$.

5.3.4. Collaborative gain: reassignments

Next to the routing costs, it is important to determine the number of reassignments that have occurred for a specific level of collaboration, which is presented in this section. The number of reassignments is expressed as the ratio between the number of reassigned requests, and the total number of initial requests in that instance. Since the collaborative penalty can not be incurred by the dynamic requests, the number of reassignments is compared with the number of initial requests instead of the total number of requests per instance.

Table 5.9: Fraction of number of reassignments for each method; Full information (FI), destroy & insertion (DI), and simple insertion (SI) and three levels of collaboration. For each method, each level is compared with the non collaborative case $\xi = 0$. Per column, the minimum and maximum are highlighted in bold. Note that highlighting is based on unrounded values.

<table>
<thead>
<tr>
<th>Method</th>
<th>FI $\xi = 1$ $\xi = 0.66$ $\xi = 0.33$</th>
<th>DI $\xi = 1$ $\xi = 0.66$ $\xi = 0.33$</th>
<th>SI $\xi = 1$ $\xi = 0.66$ $\xi = 0.33$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.61  0.48  0.28</td>
<td>0.62  0.51  0.30</td>
<td>0.61  0.48  0.33</td>
</tr>
<tr>
<td>B</td>
<td>0.52  0.34  0.23</td>
<td>0.52  0.36  0.21</td>
<td>0.56  0.36  0.26</td>
</tr>
<tr>
<td>C</td>
<td>0.69  0.55  0.34</td>
<td>0.72  0.47  0.26</td>
<td>0.72  0.47  0.24</td>
</tr>
<tr>
<td>D</td>
<td>0.64  0.61  0.34</td>
<td>0.65  0.61  0.42</td>
<td>0.62  0.58  0.50</td>
</tr>
<tr>
<td>E</td>
<td>0.67  0.51  0.33</td>
<td>0.68  0.49  0.33</td>
<td>0.70  0.49  0.33</td>
</tr>
<tr>
<td>F</td>
<td>0.55  0.40  0.12</td>
<td>0.59  0.39  0.13</td>
<td>0.59  0.36  0.11</td>
</tr>
<tr>
<td>G</td>
<td>0.66  0.55  0.26</td>
<td>0.67  0.54  0.24</td>
<td>0.69  0.52  0.23</td>
</tr>
<tr>
<td>H</td>
<td>0.65  0.52  0.23</td>
<td>0.64  0.46  0.16</td>
<td>0.63  0.45  0.14</td>
</tr>
<tr>
<td>I</td>
<td>0.63  0.54  0.29</td>
<td>0.63  0.52  0.23</td>
<td>0.61  0.47  0.29</td>
</tr>
<tr>
<td>J</td>
<td>0.77  0.67  0.26</td>
<td>0.78  0.58  0.20</td>
<td>0.78  0.59  0.15</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.64</strong>  <strong>0.52</strong>  <strong>0.27</strong></td>
<td><strong>0.65</strong>  <strong>0.49</strong>  <strong>0.25</strong></td>
<td><strong>0.65</strong>  <strong>0.48</strong>  <strong>0.26</strong></td>
</tr>
</tbody>
</table>

Discussion

Interestingly, the number of reassigned requests shows a more linear response than was the case for the routing costs. Next to that, the number of reassigned requests is much less dependent on the used solution method. For the full collaborative case $\xi = 1$, on average 65% of the total number of requests are rerouted. For high collaboration $\xi = 0.66$, the number of reassignments decreases to 50%. It is noted that decreasing the number of reassignments with 15% only increased routing costs with 1%, compared to the full collaborative case. For a low level of collaboration $\xi = 0.33$, the number of reassigned requests decreases to 26%. Trivially, for the non collaborative case $\xi = 0$ the number of reassigned requests is 0.

5.3.5. Including customer satisfaction

This section presents the results that are obtained when customer satisfaction is included. Firstly, information with respect to objective function weight $\beta_4$, the carrier quality and the
Experimental evaluation

Figure 5.14: Averaged number of reassigned requests for the three strategies full information (FI), destroy & insertion (DI), and simple insertion (SI). The number of reassignments is a fraction of the total number of requests.

customer satisfaction is presented. Secondly the obtained results, which are finally compared with the results that excluded the customer satisfaction. It is noted that no collaborative penalty is in place, \( \beta_1 = 0 \). Next to that, only the DI method is used for the cases including customer satisfaction.

Additional parameters

Prior to presenting the results, some additional parameters need to be introduced. Next to the customer satisfaction and the carrier quality, two values for objective function weight \( \beta_4 \) need to be determined.

As mentioned before, the inactivity risk is used as a measure for the customer satisfaction, with a low inactivity risk signifying a high customer satisfaction. A histogram of the distribution of the inactivity risk is shown in figure 5.15. Figure 5.15 shows that around 80% of customers are 'happy', with an inactivity risk below 10%. Furthermore, the customer satisfaction of the remaining 20% of customers is more or less equally distributed.

The values for the carrier quality are selected artificially. Again, this is because there is no carrier data available. It was decided to implement one perfect carrier, one poor carrier, and one average carrier. The respective values for each carrier are \( C_Q = 1 \), \( C_Q = 0 \), and \( C_Q = 0.5 \).

Finally, the customer satisfaction penalty is set at \( \beta_4 = 0 \) and \( \beta_4 = 10000000 \). For \( \beta_4 = 0 \) customer satisfaction is not considered in the objective function, and the objective is only to minimize routing costs. \( \beta_4 = 10000000 \) is selected, such that each customer receives the highest level of service that is needed.

Discussion

The results are shown in figure 5.16, which shows that the routing costs increase in all instances when the customer satisfaction is included. The highest increase in routing costs
is found in instance D, being 28.4%. The lowest increase is found in instance E, being 6.8%. The average result of all instances is an increase of 17.2% in routing costs.

![Figure 5.16: Routing costs for all instances excluding and including the customer satisfaction](image)

### 5.4. Concluding remarks

The solution method as presented in chapter 4 is evaluated in this chapter. The evaluation is performed by means of computational simulations of 10 instances. The instances are constructed from request data provided by the platform owner.

Three simulation methods are evaluated, which are full information (FI), destroy & insert (DI), and simple insertion (SI). These methods evaluate the instances for 4 levels of collaboration. The total set of results is used for a method comparison and a collaborative gain assessment.

The results show that the DI method outperforms SI in terms of routing cost all instances. Next to that, the collaborative gain assessment shows that marginal gains in terms of routing costs are highest for a low level of collaboration.

Lastly, including the objective of customer satisfaction is done separately. Including this additional objective results in an average increase in routing costs of +17.3%.
Conclusion

This chapter concludes this work by answering the sub questions and the main research question as stated in section 1.4, and thereby reviewing the obtained result. Furthermore, shortcomings of the conducted work are discussed and finally proposals for future research are presented.

First of all, the answers to the sub questions are given.

**SQ 1:** A mathematical model of a dynamic vehicle routing problem is constructed. It includes three objectives; minimizing routing costs, minimizing the number of reassigned requests, and maximizing the customer satisfaction. Other attributes include capacity constraints, time windows, and multiple depots. The used solution method for solving the identified VRP variation is a metaheuristic method, based on the ALNS heuristic. The method works as a centralized planner using continuous reoptimization. A centralized approach is applicable since all carriers provide complete information to the platform owner.

**SQ 2:** The routing performance is evaluated by means of a objective function that incorporates multiple objectives. Next to the routing distance and driving time, the objective function is extended with the objectives of controlling the number of request reassignments, and maintaining customer satisfaction. These three objectives are represented by means of a weighted summation of each separate objective. The weights of the objective function are chosen such that they express a monetary value.

**SQ 3:** The level of collaboration is controlled by a penalty value as part of the cost function. The reassignment of requests between carriers occurs if the marginal decrease in routing costs exceeds the collaborative penalty. The choice of a penalty value over a constraint boundary is chosen because it allows for more routing flexibility.

**SQ 4:** The solutions to the dynamic problem have worse routing performance compared to a full information problem. The routing costs are +39.3% higher on average. This is caused by two effects. Firstly, the search space for a dynamic problem is more constrained compared to a full information problem. Next to that, the dynamic nature constrains the computational time of the algorithm, although this effect was minimized in this work.

The implemented method performs better than a simple greedy method, which increases routing costs on average with +59.5%.

Also, the result of the small scale experiment is interesting to mention. The implemented method shows better performance compared to a static algorithm for the small test instance than was the case in the other test instances. Further investigation is needed to determine what the cause is for this phenomenon.
6. Conclusion

**SQ 5:** The average increase in routing costs is +17.2% when including the customer satisfaction objective. The case study, where the effect of customer satisfaction is investigated, is performed using no collaborative penalty, therefore allowing full collaboration.

This result is obtained using a high value for the collaborative penalty, so that each request receives the highest level of service needed. The increase in routing costs could be lower, at the expense of a higher risk of losing customers.

With the answers to the subquestions known, the main research question can be answered.

**RQ:** For the dynamic problems, collaborative gain is lower than for a static problem. The evaluated instances show up to 31.3% collaborative gain for the static evaluation of the instances. This is in line what was found in other literature as well. For the proposed method compared collaborative gain was 23.3% at maximum. Lowest collaborative gain was found for the simple dynamic method, where requests cannot be rerouted, being 18.3%.

Next to that, routing cost only increases with 1% when the number of reassignments is limited to 66% compared to the full collaborative case, where all requests may be reassigned. When the number of reassignments is limited to 33%, collaborative gains decrease to 13-21%.

6.1. Discussion

Although this work has obtained significant results, some shortcomings were identified.

Firstly, an improvement in the available data is preferred. As noted, synthetic data can be constructed with a bias. The used instances were only partially based on real data, since only requests that were received via the platform were available, and carrier schedules were not. It is not well known how carriers construct their schedules, with everything possible from human planning to advanced routing software. The initial schedules used in this work are all constructed using the same methodology, which may not be realistic.

Next to carrier schedules not being available, the number of requests per instance was limited, which had several effects. Firstly, the available search space is limited, especially later during the day, when a large chunk of the solution is already fixed. The limited search space also results in the algorithms scalability not known. Although not required currently, it would be disappointing if a reimplementation would be needed for larger instances.

6.2. Future research

Several opportunities have been identified for future research.

Firstly, in this work only pick up requests have been considered. Although enquiry with Quicargo employees showed that some carriers perform operations like this; separating the fleet in pick up vehicles and delivery vehicles, two other modi operandi are also interesting to consider. Firstly, from a practical point of view; VRP with backhauling. Most carriers operating in the Netherlands work using backhauling. When using backhauling, a truck is loaded with the current day’s deliveries, and when all deliveries are completed, pick ups are performed on the way back to the depot. Although the practical applicability is larger using this VRP variation, it does harm the routing flexibility by a fair bit, since delivery requests can not be reassigned between carriers (assuming each customer has unique goods).

Next to the VRP with backhauling, the VRP with pick up and deliveries (PDP), is another interesting VRP variation. The PDP variation considers the case where pick ups need to be delivered on the same day by the same vehicle. Since in this case, load does not only accumulate during a trip, but also decreases during deliveries, this may have an effect on collaborative gain in a dynamic PDP problem.

Furthermore, practical usability could be increased by including a mixed fleet. Carriers generally operate several different vehicle types, in terms of capacity/dimensions and vehicle speed. Inclusion would result in vehicle-dependent capacities and cost matrices.
6.2. Future research

Considering the collaborative penalty, in this work reassignment of initial requests between carriers results in a collaborative penalty. However, it is not well known if this actually reflects reality properly. Some other possibilities of constraining the level of collaboration are:

- Next to only incurring the penalty when initial requests are reassigned, the penalty can also be applied when a carrier initially receives a request via the platform, but later on during the day loses it again. Currently, all dynamic requests do not get an initial carrier assigned, but in this variation the first carrier that is assigned a dynamic request, becomes its initial carrier.

- Another variation would be that carriers do not mind about the specific requests that they serve, but only want to keep a certain volume. In this variation only the size of the set of initial requests is taken into account, and every request served less induces the collaborative penalty.

- The most realistic variation would probably focus on the carriers profit. With the set of initial requests and an initial schedule available a carriers initial profit can be determined. The level of collaboration can be constrained by ensuring that each carriers retains at least its initial profit, so that the decreased routing costs benefit all carriers.

The possible research directions for future research are numerous. Although the research field is already extensively studied, each different VRP variation or extension poses new challenges.


to the vehicle routing problem and its extensions: A categorized bibliography. In The
vehicle routing problem: latest advances and new challenges, pages 143–169. Springer,
2008.


2018.


[24] I. Shakil and N. Balu. Airbnb records 30% growth rate in first-quarter on booking strength:


[28] D. Lambert, M. Emmelhainz, and J. Gardner. So you think you want a partner? Mar-


cle routing systems. In The vehicle routing problem: Latest advances and new challenges,


grees of Dynamism. Technical University of Denmark. Institute of Mathematical Mod-
eling, 1996.


This appendix presents all validation results as summarized in figure 4.2.
### Table A.1: Validation results for instance set C1

<table>
<thead>
<tr>
<th>Instance</th>
<th>Algorithm</th>
<th>Best known</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vehicles [#]</td>
<td>Cost [-]</td>
<td>Vehicles [#]</td>
</tr>
<tr>
<td>c101</td>
<td>10</td>
<td>828.94</td>
<td>10</td>
</tr>
<tr>
<td>c102</td>
<td>10</td>
<td>828.94</td>
<td>10</td>
</tr>
<tr>
<td>c103</td>
<td>10</td>
<td>853.83</td>
<td>10</td>
</tr>
<tr>
<td>c104</td>
<td>10</td>
<td>1085.56</td>
<td>10</td>
</tr>
<tr>
<td>c105</td>
<td>10</td>
<td>828.94</td>
<td>10</td>
</tr>
<tr>
<td>c106</td>
<td>10</td>
<td>828.94</td>
<td>10</td>
</tr>
<tr>
<td>c107</td>
<td>10</td>
<td>828.94</td>
<td>10</td>
</tr>
<tr>
<td>c108</td>
<td>10</td>
<td>828.94</td>
<td>10</td>
</tr>
<tr>
<td>c109</td>
<td>10</td>
<td>855.20</td>
<td>10</td>
</tr>
</tbody>
</table>

### Table A.2: Validation results for instance set C2

<table>
<thead>
<tr>
<th>Instance</th>
<th>Algorithm</th>
<th>Best known</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vehicles [#]</td>
<td>Cost [-]</td>
<td>Vehicles [#]</td>
</tr>
<tr>
<td>c201</td>
<td>3</td>
<td>591.56</td>
<td>3</td>
</tr>
<tr>
<td>c202</td>
<td>3</td>
<td>749.04</td>
<td>3</td>
</tr>
<tr>
<td>c203</td>
<td>3</td>
<td>591.17</td>
<td>3</td>
</tr>
<tr>
<td>c204</td>
<td>3</td>
<td>590.60</td>
<td>3</td>
</tr>
<tr>
<td>c205</td>
<td>3</td>
<td>588.88</td>
<td>3</td>
</tr>
<tr>
<td>c206</td>
<td>3</td>
<td>588.49</td>
<td>3</td>
</tr>
<tr>
<td>c207</td>
<td>3</td>
<td>588.29</td>
<td>3</td>
</tr>
<tr>
<td>c208</td>
<td>3</td>
<td>588.32</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table A.3: Validation results for instance set R1

<table>
<thead>
<tr>
<th>Instance</th>
<th>Algorithm</th>
<th>Best known</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vehicles [#]</td>
<td>Cost [-]</td>
<td>Vehicles [#]</td>
</tr>
<tr>
<td>r101</td>
<td>20</td>
<td>1672.66</td>
<td>19</td>
</tr>
<tr>
<td>r102</td>
<td>18</td>
<td>1493.24</td>
<td>17</td>
</tr>
<tr>
<td>r103</td>
<td>14</td>
<td>1280.81</td>
<td>13</td>
</tr>
<tr>
<td>r104</td>
<td>10</td>
<td>1032.58</td>
<td>9</td>
</tr>
<tr>
<td>r105</td>
<td>15</td>
<td>1424.83</td>
<td>14</td>
</tr>
<tr>
<td>r106</td>
<td>13</td>
<td>1292.77</td>
<td>12</td>
</tr>
<tr>
<td>r107</td>
<td>11</td>
<td>1136.85</td>
<td>10</td>
</tr>
<tr>
<td>r108</td>
<td>10</td>
<td>999.98</td>
<td>9</td>
</tr>
<tr>
<td>r109</td>
<td>13</td>
<td>1209.81</td>
<td>11</td>
</tr>
<tr>
<td>r110</td>
<td>11</td>
<td>1101.90</td>
<td>10</td>
</tr>
<tr>
<td>r111</td>
<td>12</td>
<td>1091.83</td>
<td>10</td>
</tr>
<tr>
<td>r112</td>
<td>10</td>
<td>1056.04</td>
<td>9</td>
</tr>
</tbody>
</table>
### Table A.4: Validation results for instance set R2

<table>
<thead>
<tr>
<th>Instance</th>
<th>Algorithm</th>
<th>Best known</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vehicles [#]</td>
<td>Cost [-]</td>
<td>Vehicles [#]</td>
</tr>
<tr>
<td>r201</td>
<td>4</td>
<td>1270.13</td>
<td>4</td>
</tr>
<tr>
<td>r202</td>
<td>4</td>
<td>1093.93</td>
<td>3</td>
</tr>
<tr>
<td>r203</td>
<td>3</td>
<td>954.77</td>
<td>3</td>
</tr>
<tr>
<td>r204</td>
<td>3</td>
<td>827.31</td>
<td>2</td>
</tr>
<tr>
<td>r205</td>
<td>3</td>
<td>1027.78</td>
<td>3</td>
</tr>
<tr>
<td>r206</td>
<td>3</td>
<td>946.22</td>
<td>3</td>
</tr>
<tr>
<td>r207</td>
<td>2</td>
<td>946.79</td>
<td>2</td>
</tr>
<tr>
<td>r208</td>
<td>2</td>
<td>743.81</td>
<td>2</td>
</tr>
<tr>
<td>r209</td>
<td>3</td>
<td>954.67</td>
<td>3</td>
</tr>
<tr>
<td>r210</td>
<td>3</td>
<td>950.60</td>
<td>3</td>
</tr>
<tr>
<td>r211</td>
<td>3</td>
<td>821.22</td>
<td>2</td>
</tr>
</tbody>
</table>

### Table A.5: Validation results for instance set RC1

<table>
<thead>
<tr>
<th>Instance</th>
<th>Algorithm</th>
<th>Best known</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vehicles [#]</td>
<td>Cost [-]</td>
<td>Vehicles [#]</td>
</tr>
<tr>
<td>rc101</td>
<td>15</td>
<td>1688.15</td>
<td>14</td>
</tr>
<tr>
<td>rc102</td>
<td>14</td>
<td>1528.28</td>
<td>12</td>
</tr>
<tr>
<td>rc103</td>
<td>12</td>
<td>1353.80</td>
<td>11</td>
</tr>
<tr>
<td>rc104</td>
<td>11</td>
<td>1183.50</td>
<td>10</td>
</tr>
<tr>
<td>rc105</td>
<td>15</td>
<td>1602.62</td>
<td>13</td>
</tr>
<tr>
<td>rc106</td>
<td>13</td>
<td>1435.86</td>
<td>11</td>
</tr>
<tr>
<td>rc107</td>
<td>13</td>
<td>1322.19</td>
<td>11</td>
</tr>
<tr>
<td>rc108</td>
<td>11</td>
<td>1188.42</td>
<td>10</td>
</tr>
</tbody>
</table>

### Table A.6: Validation results for instance set RC2

<table>
<thead>
<tr>
<th>Instance</th>
<th>Algorithm</th>
<th>Best known</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vehicles [#]</td>
<td>Cost [-]</td>
<td>Vehicles [#]</td>
</tr>
<tr>
<td>rc201</td>
<td>4</td>
<td>1508.91</td>
<td>4</td>
</tr>
<tr>
<td>rc202</td>
<td>4</td>
<td>1236.38</td>
<td>3</td>
</tr>
<tr>
<td>rc203</td>
<td>3</td>
<td>1143.14</td>
<td>3</td>
</tr>
<tr>
<td>rc204</td>
<td>3</td>
<td>833.62</td>
<td>3</td>
</tr>
<tr>
<td>rc205</td>
<td>4</td>
<td>1322.89</td>
<td>4</td>
</tr>
<tr>
<td>rc206</td>
<td>3</td>
<td>1308.79</td>
<td>3</td>
</tr>
<tr>
<td>rc207</td>
<td>3</td>
<td>1143.42</td>
<td>3</td>
</tr>
<tr>
<td>rc208</td>
<td>3</td>
<td>907.25</td>
<td>3</td>
</tr>
</tbody>
</table>
Geographical request distribution

This appendix presents the geographical location of all requests per instance.
Figure B.1: Overview of request locations for instance A. Left side shows the initial requests. Right side shows the dynamic requests.

Figure B.2: Overview of request locations for instance B. Left side shows the initial requests. Right side shows the dynamic requests.
Figure B.3: Overview of request locations for instance C. Left side shows the initial requests. Right side shows the dynamic requests.

Figure B.4: Overview of request locations for instance D. Left side shows the initial requests. Right side shows the dynamic requests.
Figure B.5: Overview of request locations for instance E. Left side shows the initial requests. Right side shows the dynamic requests.

Figure B.6: Overview of request locations for instance F. Left side shows the initial requests. Right side shows the dynamic requests.
Figure B.7: Overview of request locations for instance G. Left side shows the initial requests. Right side shows the dynamic requests.

Figure B.8: Overview of request locations for instance H. Left side shows the initial requests. Right side shows the dynamic requests.
Figure B.9: Overview of request locations for instance I. Left side shows the initial requests. Right side shows the dynamic requests.

Figure B.10: Overview of request locations for instance J. Left side shows the initial requests. Right side shows the dynamic requests.
Results of all instances

This appendix contains all result figures.

Figure C.1: Routing costs of three strategies for instance A at varying collaborative levels

Figure C.2: Routing costs of three strategies for instance B at varying collaborative levels
C. Results of all instances

Figure C.3: Routing costs of three strategies for instance C at varying collaborative levels

Figure C.4: Routing costs of three strategies for instance D at varying collaborative levels

Figure C.5: Routing costs of three strategies for instance D at varying collaborative levels
Figure C.6: Routing costs of three strategies for instance F at varying collaborative levels

Figure C.7: Routing costs of three strategies for instance G at varying collaborative levels

Figure C.8: Routing costs of three strategies for instance H at varying collaborative levels
C. Results of all instances

Figure C.9: Routing costs of three strategies for instance I at varying collaborative levels

Figure C.10: Routing costs of three strategies for instance J at varying collaborative levels