University of Twente

Department of Industrial Engineering and Business Information Systems

Master Thesis

Improving off-line planning for vehicle routing problems in a time-dependent setting

Author:
Peter Dieter

UNIVERSITY OF TWENTE.

Supervisors:
Dr. Peter Schuur
Dr. Derya Demirtas

Supervisors:
Dr. Christian Schumacher
Johannes Blank, M.Sc.

August 18, 2020

MANAGEMENT SUMMARY

Traffic congestion represents a big problem in most road networks. During the previous decade, the total number of reported traffic congestion cases as well as the total length of those traffic congestions increased every year. In 2018, the total length of all 745,000 reported traffic congestion cases on German highways amounted to 1.5 million kilometers. Remarkable is that these numbers do not even include inner-city traffic congestion. It is estimated that in Europe every year nearly 100 billion Euros, which is corresponding to 1% of the European Union's GDP, are lost to the European economy as a result of traffic congestion. It is estimated that only 13 to 30% of the traffic congestion during peak-hours is incident related. This means that most of the traffic congestion during peak-hours is predictable and consequently, can be tackled. To reduce the impact that congestion has on route plannings, smart strategies that include time-dependent travel times have to be developed. This thesis develops and evaluates such strategies and finally answers the research question:

'How and to what extent can the Westphalia DataLab GmbH (WDL) incorporate historical traffic data to provide better routing plannings in terms of the chosen KPIs?'

In order to answer this research question, it was firstly researched what literature suggests on how historic traffic data can be used to improve off-line routing plannings. This literature review serves as the basis for the five strategies that have been developed in this research. In four of these strategies we apply the same optimization algorithm that is currently in use by the WDL (a Tabu-Search from Google's OR-tools library) and which differ just by the input to this algorithm. As only the input differs for these four strategies, we call those 'I-strategies', where the 'I' stands for input. The first I-strategy involves solving the problem with time-independent data retrieved from the OpenRoute application programming interface (API). In the second I-strategy, time-windows are compressed to prevent delays while still time-independent data retrieved from the OpenRoute API is used. In the third I-strategy, travel times are averaged over a day by using time-dependent data from Bing's distance matrix API. In the fourth I-strategy, for each directed location pair, the longest travel time (worst-case) for the specific day is determined. Also here, Bing's distance matrix API is being used. Next to that, a fifth strategy which we call 'A-strategy' has been developed. The 'A' stands for algorithm as it is a type of Simulated Annealing algorithm that takes time-dependent travel times directly into account. These strategies are then evaluated on 36 different vehicle routing problem instances, where an instance is defined as a set of customers including a depot from which the vehicles start to serve those customers. These instances were constructed from real location data in Germany. As already mentioned, the time-dependent travel time data was attained from the Bing distance matrix API. Results show that in the currently used strategy (using a time-independent travel time matrix and Google's OR-tools library) 20% of customers are served late. By simply using a travel time matrix which contains the daily average of all directed location pairs as input to the same algorithm, delays are on average reduced by more than 60% with just a slight increase of the route duration of 2%.

The A-strategy resulted in good routes for smaller instances but gave unsatisfactory results for bigger instances. These results have been validated with a second data source for time-dependent travel times, namely Google's distance matrix API. In this validation step it was also found out that the routes this A-strategy provided were less robust than the routes provided by other strategies. Overall, we recommend the Westphalia DataLab to include historic travel time data to improve the route plannings they provide to their clients. More specifically, the following suggestions are made:

- Bing's distance matrix API (or an equivalent API) should be used to attain time-dependent travel times as it provides generous API request limits and travel time estimations whose quality is similar to more expensive options such as Google's distance matrix API.
- Generally, given a vehicle routing problem for a specific day, a travel time matrix which contains the daily average of that specific day of all directed location pairs should be applied in combination with the currently used optimization algorithm.
- In cases where delays must absolutely be avoided, a travel time matrix which contains the longest travel time (worst-case) for the specific day for each directed location pair should be applied in combination with the currently used optimization algorithm.
- Further research that investigates other strategies to incorporate time-dependency into routing plannings should be conducted. Special attention should be paid to the robustness of these strategies.

When following these recommendations, we expect the Westphalia DataLab to advance their routing capabilities and ultimately provide a higher value to their clients.

PREFACE

This thesis marks the end of my master studies and there are many people without whom I would not have achieved this.

Firstly, I would like to thank my first university supervisor Peter Schuur for guiding me through this work and providing me very valuable feedback. It was a pleasure to work with you. Also, I would like to thank my second university supervisor Derya Demirtas for her feedback and clear remarks that substantially improved this work. Secondly, I want to thank Christian Schumacher and Johannes Blank from the Westphalia DataLab. Not only your remarks but especially your creative ideas have helped me a lot. Even though we were separated due to the Corona pandemic it was very smooth to work with you and I enjoyed the time at the WDL. Thirdly, I want to thank all the teachers and friends from all around the world who I have met during my great time as a student. I am very grateful that I had the possibility to study in three amazing countries: the Netherlands, France and Portugal. Finally, I want to thank my parents, my brother and my sister who have supported me throughout the study.

I am looking forward to a future in which I can apply the broad set of skills which the study has provided to me while always continue learning and exploring.

ABBREVIATIONS

AETR European Agreement concerning the work of crews of vehicles engaged in international Road Transport. 19

API application programming interface. i, 2, 3, 14, 20–22, 24, 31, 32, 34, 39

CoV Coefficient of Variation. 37–39

CVRP Capacitated Vehicle Routing Problem. 5, 8, 11

CVRPTW Capacitated Vehicle Routing Problem with Time Windows. 23, 24

DCVRP Distance and Capacity constrained Vehicle Routing Problem. 8

FIFO First in - First out. 12, 13, 22

GRASP Greedy Randomized Adaptive Search Procedure. 14

ILP Integer Linear Program. 6, 7, 12, 14

KPIs Key Performance Indicators. 1–4, 15, 17, 29–31, 40, 43, 54, 57

MILP Mixed-Integer Linear Program. 12

STDVRP Stochastic Time Dependent Vehicle Routing Problem. 14

TDCVRPTW Time Dependent and Capacitated Vehicle Routing Problem with Time Windows. 18, 24

TDVRP Time Dependent Vehicle Routing Problem. 12–15

TSP Travelling Salesman Problem. 5–7, 27

VRP Vehicle Routing Problem. 1, 4–13, 15, 17–19, 40, 57, 58

VRPB Vehicle Routing Problem with Backhauling. 8

VRPPD Vehicle Routing Problem with Pick-up and Delivery. 8

VRPTW Vehicle Routing Problem with Time Windows. 5, 8, 11

WDL Westphalia DataLab GmbH. i, viii, 1–5, 8, 20, 23, 24, 31, 32, 34, 56–58

GLOSSARY

client A client of the Westphalia DataLab.

customer A person or company that needs to receive a delivery from a client.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER
ONE

INTRODUCTION

I am writing this thesis in the framework of my master studies in 'Industrial Engineering and Management' with a specialisation in 'Production and Logistics Management' at the University of Twente. It is conducted at the WDL in Münster (Germany) and investigates the use of traffic data in Vehicle Routing Problems. In this chapter, firstly the company is presented briefly (Section 1.1). Secondly, the current work flow of routing projects is described (Section 1.2). Thirdly, the problem cluster including the core problem is identified (Section 1.3). Based on this problem description, research questions are proposed and the goal of the study is defined (Section 1.4). Lastly, the research approach and the respective structure of the thesis that is followed throughout the report is presented (Section 1.5).

1.1 Company Description

The WDL, located in Münster (Germany), is a software manufacturer which was founded in 2017 with the goal to provide AI powered solutions to their clients. As of March 2020, the company employs almost 70 people. The company is active in different areas in which large amounts of data have to be processed. This includes for instance demand forecasting or the analysis and interpretation of medical images. Next to these traditional data analysis tasks, the company is also offering Operations Research services. This predominantly includes providing solutions to variants of the Vehicle Routing Problem (VRP) which consists of "designing least cost delivery routes through a set of geographically scattered customers, subject to a number of side constraints" (Laporte et al., 2013).
The services offered by the WDL can be split up into two different types: standard software and tailored software. As the name suggests, standard software can be used by a broader range of clients without the need of further modifications. An example for this is 'X-4C', a forecasting tool for time series data. On the other hand, tailored software is software that is created from the ground up or strongly modified, specific to the clients' needs. So far, most routing projects required these kind of modifications and can therefore be classified as tailored software. Usually, customers have very particular requirements including multiple transportation types (e.g. transportation by truck and train) and other constraints.

1.2 Routing Project Workflow

To give an impression about the process of routing projects, the current work flow of routing projects is presented in Figure 1.1. As it is usually the case for tailored software, every project starts with a request of the client to perform a project. This client request includes matching the clients' requirements, including the definition of relevant Key Performance Indicators (KPIs), with WDL's capabilities. If

consensus is found, the client provides relevant and problem specific data and WDL's operations research team is starting to formulate the problem in a structured and analytical way. After the problem is formulated and the client provided his data, the data will be processed in such a way that it can be fed into an algorithm/ different algorithms. This data processing step also includes retrieving external data, such as distance matrices which are currently attained by the 'Openrouteservice' API (*OpenRouteService Matrix*, 2020). Existent algorithms such as those from Google OR-tools (Perron & Furnon, 2019) are applied, which are modified to the current problem. These routing algorithms are then run, tested, and the most appropriate one, based on the chosen KPIs, is implemented into the client's software.

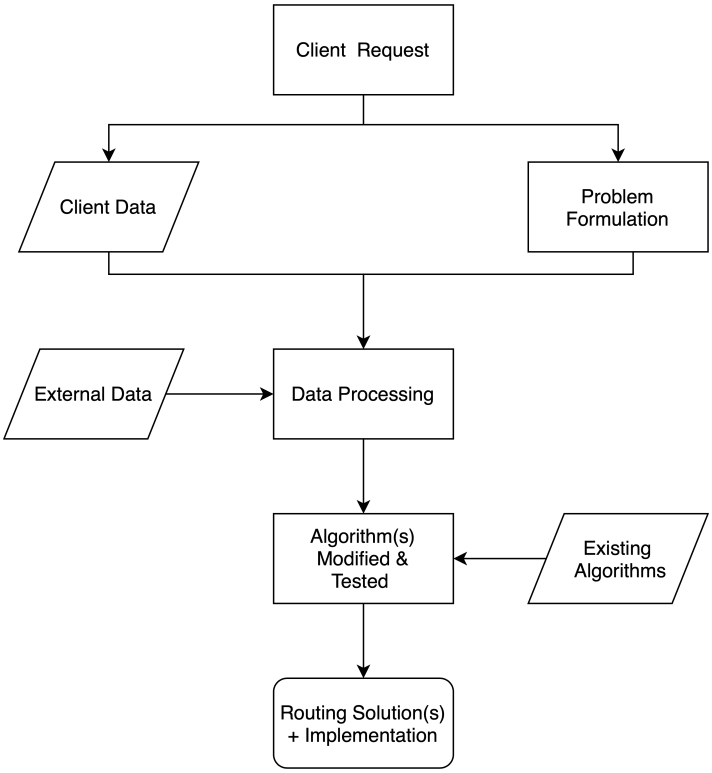


Figure 1.1: Work Flow of Routing Projects

1.3 Problem Analysis

Traffic congestion represents a major problem in many road networks. In Germany, the total number of reported traffic congestion cases as well as the total length of those traffic congestions increased every year during the previous decade. In 2018, the total length of all 745,000 reported traffic congestion cases on German highways amounted to 1.5 million kilometers. While these numbers only present the problem on highways, also inner-city traffic, especially in densely-populated areas, is prone to congestion in peak hours when people go or return from work (Mancini, 2014). This problem is becoming more acute with higher utilization of road networks due to an increasing number of delivery vehicles as a result of more online shopping. Traffic congestion leads to longer routes (as measured in time) and delays on the customers' site. This is especially an issue for industries where delays pose a big problem, such as hospitals. Next to that, also fuel-consumption is higher in traffic congestions, leading to higher costs, financially and environmentally. Savelsbergh & Van Woensel (2016) estimate that in Europe every year nearly 100 billion Euros, which is corresponding to 1% of the European Union's

(EU's) GDP, are lost to the European economy as a result of traffic congestion. Traffic congestions have different causes. Some causes (and the resulting congestion) can be well predicted, such as high traffic intensity in peak times. Other causes are less or even unpredictable, such as accidents or bad weather conditions that force the traffic to slow down. According to Skabardonis et al. (2003), only 13 to 30% of the traffic congestion during peak-hours is incident related. This means that most of the traffic congestion during peak-hours is predictable and consequently, can be tackled.
The above described issue also raises problems for the WDL. The algorithms currently used by the WDL do not take historic traffic into account when determining the suggested routes. This can lead to a resulting problem, namely sub-optimal routing solutions which could be improved with a reasonable and economical amount of effort. As a major goal of the operations research team is to attain as good as possible routing solutions within the scope of the project, sub-optimal routing solutions represent an urgent problem. It is possible that shorter routes that do not violate any constraints are omitted and next to that, delays at a customer can result from wrong estimations of the travel time. Two currently unknown factors play an important role in solving this issue. Firstly, it is not known if solutions would be improved if historic traffic data would be accounted for. It is possible that incorporating historic traffic data would not lead to (significantly) better routing solutions but would rather over-complicate an already difficult project. Secondly, it is currently not known how to incorporate historic traffic data into algorithms. Multiple strategies with different levels of complexity exist and could possibly be incorporated into the applied algorithms.

1.4 Motivation & Research Questions

In recent years, multiple routing data providers such as 'Google Maps', 'Bing Maps' and 'Openrouteservice', offer APIs to private persons and businesses. These are programming interfaces that make it possible to retrieve predicted travel times. Services such as Google or Bing Maps incorporate historical traffic data whereas other providers including 'Openrouteservice' do not. Furthermore, providers who incorporate historical traffic data make it possible to specify a date and the time of the day when the travel is supposed to start (time-dependent travel times) while other providers including 'Openrouteservice' just provide time-independent travel times. This development makes it feasible to include historical traffic data in a feasible way. As 70 to 87% of traffic congestion is predictable (Skabardonis et al., 2003), the focus of this research is on this accountable part. We therefore deal with off-line planning, that is, the planning of routes before they physically start. There are several strategies to deal with this predictable type of traffic congestion which will be mentioned later. The goal of this research is to evaluate the degree to which historical traffic data should and could be incorporated into routing plannings provided by the WDL to its clients. This should result in better routing solutions in terms of the KPIs chosen. We therefore formulate the following main research question:

'How and to what extent can the WDL incorporate historical traffic data to provide better routing plannings in terms of the chosen KPIs?'

The main research question contains multiple aspects that should be tackled separately. This is done by narrowing down the main research question into multiple sub-questions. Therefore, the following sub-questions are posed and answered on corresponding chapters:

Literature Review (Chapter 3)

1. What does literature suggest on how historic traffic data can be used to improve off-line routing plannings?

Methodology (Chapter 4)

2. Which research framework can be set up to test different solving strategies for the VRP that incorporate historic traffic data?
3. Which data can be used to model realistic, time-dependent VRP instances?
4. Which strategies to incorporate historic traffic data should be tested?
5. Which KPIs could be used in order to compare the different strategies?

Data Exploration (Chapter 5)

6. Which characteristics does the collected travel time data have and which influences do these characteristics have on the experimental design?

Experiments (Chapter 6)

7. How can the collected data be used to model realistic VRP instances?
8. To what degree does predictable traffic intensity influence current routing plannings in terms of the chosen KPIs?
9. How do the tested strategies perform in terms of the chosen KPIs?
10. How can the results be validated?

Conclusion and Recommendations (Chapter 7)

11. Which strategies are most favorable to put into practice by the WDL?

1.5 Research Approach

As outlined, the report is structured by providing answers to the sub-questions of this research, which will guide us to an answer of the main research question.
Before the first sub-question will be answered, background knowledge about different variants of the VRP is presented to provide the reader basic knowledge about the general type of problem we are dealing with in this research. This is done in the second chapter. In the third chapter, the first sub-question is answered by conducting a literature review on how historic traffic data can be incorporated into routing plannings. Chapter 4 includes the methodological design of this research. Firstly, a research framework is set up. Afterwards, the data which is used is presented. This includes the presentation of the location data as well as the travel time data used in this research. Thirdly, different strategies to include time-dependent travel times are developed and described in detail. Lastly in Chapter 4, the KPIs which will be looked at in this study are presented. In Chapter 5 an exploration of example data is performed in order to see which characteristics the collected travel time data has and how these characteristics should be incorporated in the experimental design. Chapter 6 includes the construction and execution of experiments. Next to that, also the results of those experiments and their validity will be presented. This will then result in the answer to the main research question, namely how and to what extent the WDL can incorporate historical traffic data to provide better routing plannings. This will be done in Chapter 7, the conclusion and recommendations.

CHAPTER
TWO

BACKGROUND KNOWLEDGE

This chapter provides an introduction to different types of VRPs. We start with the definition of the simplest VRP form, the travelling salesman problem (Section 2.1). We then introduce the classical VRP with multiple vehicles. Next to that, two common variants of the VRP with which we deal later in the report, namely the Capacitated Vehicle Routing Problem (CVRP) and the Vehicle Routing Problem with Time Windows (VRPTW), are presented (Section 2.2). The reason why we look at those are that these two variants are the basis for many other extensions of the classical VRP and both are commonly used in practice, also by the WDL. The chapter ends with the introduction of the most common solving methods for the VRP (Section 2.3).

2.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is one of the most famous combinatorial optimization problems in the fields of Computer Science and Operations Research. The problem is stated as follows: 'Given a set of cities along with the cost of travel between each pair of them, (...) find the cheapest way of visiting all the cities and returning to the starting point' (Applegate et al., 2006). The TSP can be seen as the simplest form of a VRP. There is only one depot, one vehicle with unlimited capacity and no further constraints exist. A TSP with seven cities is visualized in 2.1 , where the bottom (green) square is the starting city.

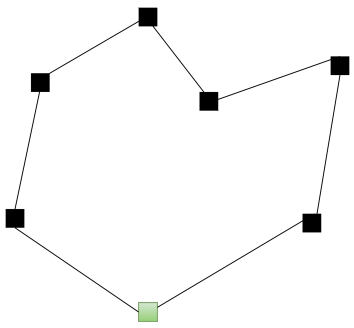


Figure 2.1: Visualization of a Travelling Salesman Problem

The TSP is a NP-hard problem. That means, that the problem can probably not be solved in polynomial time. Given N cities (including the same start and end city) and a symmetric distance matrix, there exist $\frac{(N-1)!}{2}$ feasible possibilities of visiting every city once and returning to the starting

city. For example, with 13 cities, there are already more than 3 billion possible routes. The TSP can be formulated as the following Integer Linear Program (ILP). Where N are the number of cities and x_{ij} is 1 if there is a path from city i to city j and 0 otherwise. The distance between city i and city j is given by c_{ij}. The variable u_i is a dummy variable that is needed for the subtour-elimination constraint. While x_{ij} and u_i are decision variables, N and c_{ij} are parameters of the ILP that have to be given beforehand.

$$\text{minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_{ij}$$

$$\text{subject to} \quad \sum_{i=1,i\neq j}^{N} x_{ij} = 1, \qquad j = 1,...,N \tag{2.1}$$

$$\sum_{j=1,i\neq j}^{N} x_{ij} = 1, \qquad i = 1,...,N \tag{2.2}$$

$$u_i - u_j + Nx_{ij} \leq N-1, \quad 2 \leq i \neq j \leq N \tag{2.3}$$

$$0 \leq u_i \leq N-1, \quad 2 \leq i \leq N \tag{2.4}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 1,...,N \tag{2.5}$$

$$u_i \in \mathbb{Z}, \quad i = 2,...,N \tag{2.6}$$

While the objective function makes sure that all paths travelled are included as costs, equation 2.1 and 2.2 assure that each city is visited and left exactly once. Equations 2.3 and 2.4 are also known as the Miller-Tucker-Zemlin formulation that prevent possible sub-tours. The logic is as follows: each city except the starting city (city 1) gets a dummy variable u_i assigned, sequentially in form of an integer between 0 and $n-1$. By equation 2.3, if there is a path from city i to city j, then the dummy variable of city j must be larger than the one of city i. This prevents returning to an already visited city except it is the starting city, because no dummy variable u_1 was assigned to the starting city yet. When arriving at the last city visited, the only possibility to not break the constraint is to return to the starting city. Therefore, u_i will result in an ordered sequence that represents the sequence of cities visited. Constraints 2.5 and 2.6 are integrality conditions.

2.2 Vehicle Routing Problem

2.2.1 Standard Vehicle Routing Problem

As stated in Chapter 1, the VRP is defined as "designing least cost delivery routes through a set of geographically scattered customers, subject to a number of side constraints" (Laporte et al., 2013). Instead of visiting cities as it is the interpretation of the TSP, customers with a certain demand have to be served in the VRP. Another key difference to the TSP is that usually the problem involves more than one available vehicle that starts from a depot. Figure 2.2 shows a VRP in which three vehicles are used.

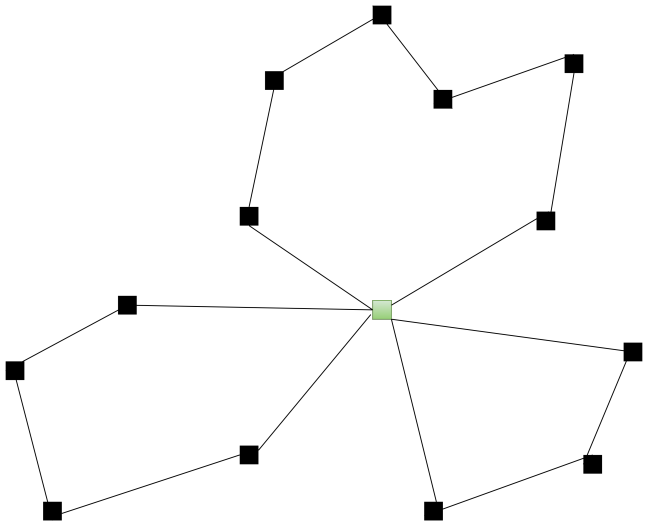


Figure 2.2: Visualization of a Vehicle Routing Problem

As it is the case for the TSP, also the VRP is a NP-hard problem. It can be formulated as the following ILP as adapted by Christofides et al. (1981). There are now M vehicles, N customers with the depot on node 0. The decision variable x_{ijk} is 1 if vehicle k travels from node i to node j and 0 otherwise. The remaining variables are similar to the one of the TSP formulation.

$$\text{minimize} \quad \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=1}^{M} c_{ij}x_{ijk}$$

$$\text{subject to} \quad \sum_{i=0}^{N}\sum_{k=1}^{M} x_{ijk} = 1, \qquad j = 1, ..., n \tag{2.7}$$

$$\sum_{i=0}^{N} x_{ipk} - \sum_{j=0}^{N} x_{pjk} = 1, \qquad k = 1, ..., M, \quad p = 0, ..., N \tag{2.8}$$

$$\sum_{j=1}^{N} x_{0jk} = 1, \qquad k = 1, ..., M \tag{2.9}$$

$$u_i - u_j + N\sum_{k=1}^{K} x_{ijk} \leq N - 1, \quad 1 \leq i \neq j \leq N \tag{2.10}$$

$$0 \leq u_i \leq N - 1, \quad 1 \leq i \leq N \tag{2.11}$$

$$x_{ijk} \in \{0, 1\}, \quad i, j = 0, ..., N \;\; k = 1, ..., M \tag{2.12}$$

$$u_i \in \mathbb{Z}, \qquad i = 1, ..., N \tag{2.13}$$

Equation 2.7 makes sure that every customer is visited exactly once. 2.8 makes sure that if a vehicle visits a node it must also leave this node. Equation 2.9 states that every vehicle leaves the depot exactly once. Equation 2.10 and 2.11 are the sub-tour elimination constraints as explained in Section 2.1. Constraints 2.12 and 2.13 are the integrality conditions.

Many variants of the VRP with different side constraints exist. Figure 2.3 shows a scheme of the most common variants as presented by Toth & Vigo (2001).

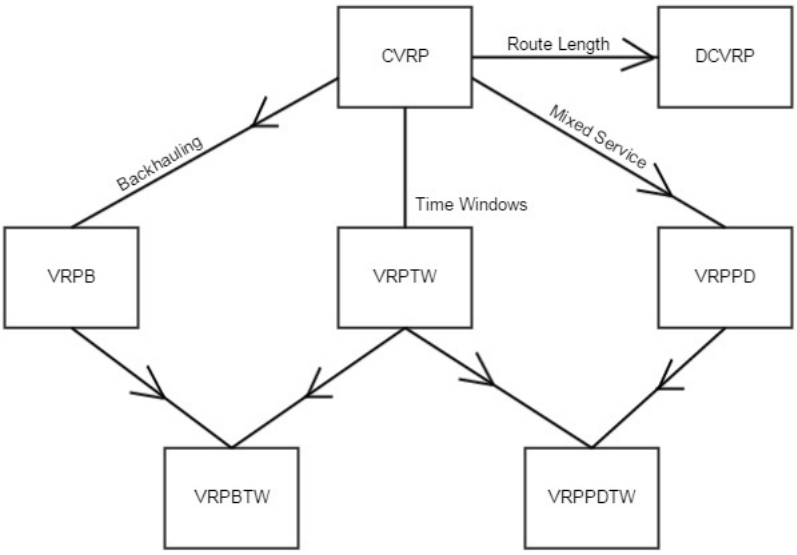


Figure 2.3: VRP Variants as described by Toth & Vigo (2001)

From the scheme we can see that the CVRP and the VRPTW are the basis of further problems. These two variants constrain the truck to have a certain maximum capacity and to arrive at the customer's site in a certain time window, respectively. In the Distance and Capacity constrained Vehicle Routing Problem (DCVRP), the distance that each vehicle is able to travel is limited. The Vehicle Routing Problem with Pick-up and Delivery (VRPPD) includes a set of vendors whose goods need to be transported back to the depot, next to delivering customers. The Vehicle Routing Problem with Backhauling (VRPB) is similar with the restriction that all deliveries for a route must be completed before any pickups are made. In order to assure generality of the research for future projects of the WDL, only the CVRP and the VRPTW will be addressed further on. We continue with introducing these two variants in more detail.

2.2.2 Capacitated Vehicle Routing Problem

As stated above, the Capacitated Vehicle Routing Problem (CVRP) is a variant of the VRP in which the vehicles have a maximum capacity that cannot be exceeded. Whenever the vehicle has to deliver certain kind of physical goods and not just a service is offered, a capacity constraint has to be added. Equation 2.14 is an example of a possible capacity constraint that can be added to the formulation in Subsection 2.2. The maximum capacity for each vehicle is given by Q and the demand of customer i is given by q_i with $q_0 = 0$ (no demand at the depot). As Q does not change per vehicle, the constraint is addressing a homogeneous fleet.

$$\sum_{i=1}^{N} \sum_{j=0}^{N} q_i x_{ijk} \leq Q, \quad k = 1, ..., M \tag{2.14}$$

2.2.3 Vehicle Routing Problem with Time Windows

In the Vehicle Routing Problem with Time Windows (VRPTW) a vehicle can only arrive at a node during a given time window. It represents opening hours of customers in real-life. A vehicle can arrive earlier but has to wait (with no waiting costs) until the customer is opening. In order to model this constraint in an ILP form, the following two constraints can be added to the ILP in subsection 2.2. The decision variable y_i represents the time that we arrive at node 0. The opening and closing time at node i are given by a_i and b_i respectively. M is a sufficiently large number.

$$y_i + c_{ij} - y_j \leq M(1 - x_{ijk}), \quad i, j = 0, ..., N \quad k = 1, ..., M \tag{2.15}$$

$$a_i \leq y_i \leq b_i, \qquad i = 0, ..., N \tag{2.16}$$

Please note that in the equation above we assume that the time distance between node i and node j is relative to the cost of travelling. If this is not the case, we would have to replace c_{ij} in 2.15 with t_{ij}, the time it takes to travel from node i and node j.

2.3 Solving Methods

All the problems mentioned above are solvable by exact methods as well as by metaheuristics. While exact methods are just applied to solve relatively small problems due to complexity (time) reasons, metaheuristics are meant to find 'good enough' solutions, not the optimal one. Metaheuristics are often nature inspired algorithms which include mechanisms that avoid getting trapped in local optima (Laporte, 2009). We will now shortly present the basis of the most common algorithms that are used in literature and practice to solve the VRP as identified by Braekers et al. (2016).

2.3.1 Exact Methods

Branch-and-Bound

The branch-and-bound algorithm can be defined as a state space search in form of a tree with the full set of solutions at its root. The solution is therefore being built gradually. In the classical depth-first approach, at every node of the tree there are two steps to be performed, bounding and further branching:

1. A solution to the LP-Relaxation of the problem at the node is attained (e.g. with the simplex method). If no feasible solution is found, than the branch can be cut and not further be examined. If a feasible and integer solution is found, than the branch does not have to be examined further because it is rather worse than the current best solution or it is the best solution found so far and further branching on this node would be redundant.
   In these two cases step two will be performed on a different node which has to be selected according to a predefined rule.
   If a feasible non integer solution is found, than step two will be performed on the current node.

2. New branches are added by gradually building up a solution from the current node. This step is called branching.

The optimum is found if there are no nodes in the tree that result in a non-integer solution and have not been further branched upon yet.

Branch-and-Cut

The branch-and-cut algorithm has proven to be more efficient than the branch-and-bound algorithm for most problems including the VRP. It combines branch-and-bound and cutting planes for the LP-relaxation. These cutting planes are constraints that have to be satisfied by all feasible integer solutions while giving a solution that is closer to the best feasible integer solution. Better bounds are therefore attained that help in cutting the tree in a more efficient way.

2.3.2 Heuristics

There are two main types of heuristics: Constructive heuristics and improvement heuristics. While in the first one the goal is to find a good feasible solution from scratch, the goal of improvement heuristics is to improve a given solution.
A sub-type of constructive heuristics is called 'divide and conquer'. This is a heuristic in which the problem is split into smaller sub-problems which are then solved (often to optimally with exact methods). The solutions to those sub-problems are then used to (re-)construct an overall solution to the entire problem.
An example of improvement heuristics are local search methods. These define neighboring solutions of the current solution beforehand and accept to move to a neighbor if the solution improves. If no neighbor with a better solution exists, the algorithm stops. These local search algorithms get likely stuck in a local optimum because they are not able to move to all possible solutions, given the same start solution. More successful improvement heuristics are global search algorithms. These do not only explore neighboring solutions but are also able to move to solutions that are 'further away' (i.e. more and/or different types of moves are needed) by incorporating the possible acceptance of worse solutions. We now explain the most common global search algorithms in more detail: Simulated Annealing and Tabu Search.

Simulated Annealing

Simulated Annealing is a nature inspired algorithm. It was first introduced by Kirkpatrick et al. (1983). The name annealing comes from the field of metallurgy in which it describes the "treatment of a metal or alloy by heating to a predetermined temperature, holding for a certain time, and then cooling to room temperature to improve ductility and reduce brittleness" (*Encyclopædia Britannica*, 2011). In an optimization context, we start with a given solution and then accept to move to a worse but feasible solution with a certain probability, which is an equivalent to a temperature and will be further referred to as such. Like in annealing with metallurgy, the temperature decreases with each iteration. Moving to an improved and feasible solution is always accepted. The best feasible solution found so far is usually stored. The algorithm begins with a given start solution and a predefined neighboring structure. The algorithm stops when a maximum number of iterations is reached or when an other stopping criterion is met. Multiple cooling schemes for the temperature exist such as linear or stepwise cooling schemes. Most commonly however, cooling schemes in some negative exponential form are applied. Next to that, one can also allow the algorithm to temporarily move to infeasible solutions by incorporating some form of penalty if certain constraints are not met. This is especially useful if different sub-spaces of the entire (feasible) solution space are not connected by the given neighborhood structure. The following expression is an example of a negative exponential cooling scheme where T_k is the temperature at iteration k and c is a cooling factor that needs to be chosen beforehand but should be sufficiently small as otherwise the temperature would decrease too fast and not much exploration would be performed.

$$T_k = e^{-kc} \tag{2.17}$$

Tabu search

The Tabu search algorithm was firstly introduced by Glover & McMillan (1986) for scheduling optimization. As for Simulated Annealing, a start solution is needed beforehand as well as a neighboring structure for solutions. Besides that, an initially empty tabu list with a predefined length is introduced. The aim of the list is to prevent cycling between the same solutions without exploring much of the solution space. This list either includes entire solutions or, more commonly, certain moves that are part of the neighboring structure which have been performed before. At every iteration a set of neighboring solutions is determined and we move to the one with the best objective value, if the move is not part of the tabu list. As the tabu list has a certain length, the performed move is added to the tabu list and in case that the tabu list is full (i.e no. of iterations $\geq$ length of tabu list), the first entry of the tabu list is deleted. The algorithm stops when either a certain convergence criteria is met or after a maximum number of iterations.

2.4 Conclusion

The VRP is a widely studied problem type in operations research which consists of "designing least cost delivery routes through a set of geographically scattered customers, subject to a number of side constraints" (Laporte et al., 2013). Many variants of the VRP exist such as the CVRP and the VRPTW, which include capacity constraints of the vehicles used and constraints for the visiting time at a customer's site, respectively. The TSP is a NP-hard problem, the problem can therefore probably not be solved in polynomial time. Consequently, exact methods such as branch-and-bound or branch-and-cut can only be used to solve sufficiently small problems. Simulated annealing and Tabu-Search are two types of global search improvement heuristics. These types of heuristics are designed to find 'good enough' solutions in a given amount of time.

CHAPTER

THREE

LITERATURE REVIEW

This chapter answers the following research question: *'What does literature suggest on how historic traffic data can be used to improve off-line routing plannings?'*. This will be done by conducting a literature review on previously applied or suggested strategies. In general, the research can be split up into two parts: deterministic and stochastic travel times (Section 3.1 and 3.2, respectively). While this literature review includes both cases, the focus will be on the deterministic case as the data retrieved is of time-dependent, yet deterministic nature.

3.1 TDVRP with Deterministic Travel Times

In order to achieve a more realistic representation of real-life conditions, Malandraki & Daskin (1992) suggested and established the term Time Dependent Vehicle Routing Problem (TDVRP). The TDVRP extends the classical VRP to account for traffic congestion. While the VRP assumes constant travel times, the TDVRP incorporates time-dependencies into their travel times, depending on the hour of the day when the travel starts. Next to introducing the term, Malandraki & Daskin (1992) formulated a Mixed-Integer Linear Program (MILP) for the TDVRP. They also suggest multiple nearest-neighbor heuristics and a branch-and-cut algorithm which are tested on small and randomly generated instances. The branch-and-cut algorithm results in better solutions in more than 2/3 of the cases, compared to the suggested nearest-neighbor heuristics. Their objective function minimizes the total route time of all vehicles (travelling, service and waiting time). The way Malandraki & Daskin (1992) incorporated travel times is that they include a new constant c_{ij}^m which is defined as the travel time from node i to node j if starting at i during time interval m. From this formulation we can see that travel times are computed using simple step functions. This however results in not satisfying the First in - First out (FIFO) or also called 'non-passing property' on travel times: It can happen that, starting in point A, vehicle i would arrive earlier at point P if it starts the travel later. This can be visualized in Figure 3.1. If leaving at 11:59, the travel time would be 30 minutes and we would arrive at 12:29. If we leave at 12:00 sharp, the travel time is 10 minutes and we would arrive at 12:10. In this specific case, we would therefore arrive 19 minutes earlier if we leave one minute later. This does not reflect real life, because when taking the same route, an earlier start should also result in an earlier arrival.

Jung & Haghani (2001) satisfy the FIFO requirement by applying a continuous travel time function. Next to that, also an ILP formulation and a genetic algorithm for the TDVRP is presented. Experiments showed that for small problems, the genetic algorithm results in the optimal solution in 31 out of 33 cases with a maximum optimally gap of 5% in the other two cases while consuming considerably less time. For larger problems, the largest optimally gap between the genetic algorithm and the optimal solution is less than 7%. Ichoua et al. (2003) solve the FIFO requirement by intro-

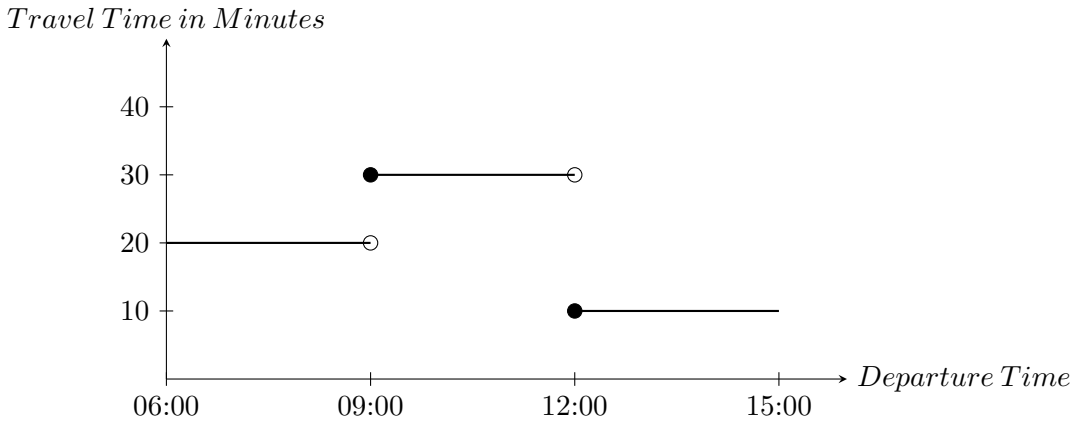


Figure 3.1: Travel Time Step Function

ducing speed step functions instead of travel time functions. Looking at Figure 3.1, this would mean that instead of the travel time, we would have the travel speed on the y-axis and not the departure time, but just the current time on the x-axis. We assume that the shown speed function holds for the 30km long arc from city i to city j and travel speed is measured in km/h. If we start at 08:57, we would travel 20km/h for three minutes and 30km/h until we reach our destination. We would therefore arrive at 09:58 with a total travel time of one hour and one minute. If starting at 9:00, we would travel constantly with a speed of 30km/h and would therefore arrive at 10:00, with a total travel time of one hour. Even though the travel time is shorter when starting later, we would still arrive at a later time. In fact, we would never arrive at an earlier time when leaving later. The FIFO property is therefore satisfied. Ichoua et al. (2003) also suggest a parallel tabu-search heuristic and show the time-dependent model provides substantial improvements over a model based on fixed travel times. Donati et al. (2008) used a multi-ant colony system to solve the VRP and extended this system to solve the TDVRP. The algorithm is tested on a model derived from data of a real-world road network. This model uses speed step functions to compute the travel time between destinations. The algorithm with time-dependent travel times as input improves the solution found by the same algorithm with time-independent travel time as input by almost 8%. Osvald & Stirn (2008) applied a tabu-search algorithm to solve the TDVRP in an environment with perishable goods (such as fresh vegetables). The algorithm was tested on various problem instances which were derived by analyzing food wholesalers in Slovenia. They show that their algorithm results in a 47% reduction of perished goods. Balseiro et al. (2011) present an improved ant-colony optimization algorithm by hybridizing it with insertion heuristics. Their algorithm first minimizes the number of routes and then the total time. The resulting algorithm matches or improves the best known results in several benchmark problems. In the paper of Kok et al. (2012), the authors tested four different strategies. All these strategies use a restricted dynamic programming algorithm presented by Gromicho et al. (2012) which can take both dependent as well as time-independent travel times as input. The algorithm firstly minimizes the number of vehicles used and then minimizes the total travel duration. A speed step model and problem instances are presented which are based on real-life data from areas in the US. The first strategy serves as a base-line strategy by ignoring time-dependency by setting the speed of each arc to its maximum. The second strategy accounts for traffic congestion by setting the speed of the arcs to their averages. The third strategy avoids traffic congestions by using time-dependent travel times as input to the above mentioned algorithm. While the third strategy computes the shortest path with time-independent travel times, the fourth strategy also takes time-dependent travel times into account to compute the shortest path between two locations. Also the fourth strategy then applies the algorithm presented by Gromicho et al. (2012) to solve the TDVRP. The strategies are summarized

Table 3.1: Strategies by Kok et al. (2012)

Strategy	*Shortest Paths*	*Travel Times input for VRP*	*Accounting for congestions*	*Avoiding congestions*
1	Time-indep	Time-indep	No	No
2	Time-indep	Time-indep	Yes	No
3	Time-indep	Time-dep	Yes	Yes
4	Time-dep	Time-dep	Yes	Yes

by Kok et al. (2012) as shown in Table 3.1. Compared to the base strategy (strategy 1), strategies 2-4 result in more robust travel schedules as the number of late arrivals and the total late time decreases drastically. More specifically, strategy 3 and 4 result in around 99%, and strategy 2 in around 70% less late arrivals then the base strategy. However, especially for strategy 2 this robustness comes with the use of more vehicles and a higher total duty time.

Dabia et al. (2013) presented an exact algorithm for the TDVRP that minimizes the total route duration. The proposed algorithm is tested on the Solomon instances which were modified by the authors to account for time-dependency. Within a computing limit of four hours, about 63% of the instances with 25 customers, 38% of the instances with 50 customers, and 15% of the instances with 100 customers were optimally solved. Lombard et al. (2018) used a Greedy Randomized Adaptive Search Procedure (GRASP) to solve the TDVRP with the objective to minimize the total route duration. They tested the algorithm on small problem instances in the city of Paris. These problem instances were modeled with real-life data from the Google Distance Matrix API. Two strategies were compared and solved with the GRASP algorithm: The first one just takes the average of the time-dependent travel time matrix as an input while the second strategy takes the time-dependent travel times as input. On average, the second strategy, which takes time-dependent travel times into account, results in a travel time duration decrease of 4.3% compared to the first strategy.

3.2 TDVRP with Stochastic Travel Times

In the previous section we have seen how time-dependent travel times can be applied to improve vehicle routing plannings. All the previous seen literature studied time-dependent yet deterministic traveltimes. In reality however, travel times are not deterministic but stochastic. Nahum & Hadas (2009) defines the Stochastic Time Dependent Vehicle Routing Problem (STDVRP) and introduces an ILP formulation for the problem. Next to that, also a constructive heuristic is proposed which is a variant of the savings-algorithm by Clarke & Wright (1964). They tested the algorithm on 190 scenarios and attained on average solutions whose objective values were 10% higher than the optimal solutions. Lecluyse et al. (2009) created different theoretical scenarios and tested a tabu-search algorithm that includes the standard deviation of the travel time in the objective function. Their experiments showed that while route duration increases to a small amount, the robustness of routes increases to a large amount, resulting in more reliable routes. Their model does not include any time-windows. Tas et al. (2014) developed two meta-heuristics, namely a Tabu Search and an Adaptive Large Neighborhood Search to solve the stochastic TDVRP with soft time-windows. The objective of their model includes a weighted combination between service costs due to early arrivals as well as delays, and transportation costs resulting from distance, overtime and the number of vehicles used. The algorithms performed well on a stochastic and time-dependent version of the Solomon instances.

3.3 Conclusion

In this chapter we answered the following research question: *'What does literature suggest on how historic traffic data can be used to improve off-line routing plannings?'* by conducting a literature review on the TDVRP. While the real life environment is stochastic, due to data restrictions this research solely focuses on deterministic travel times. The TDVRP is a variant of the VRP that was introduced by Malandraki & Daskin (1992) to account for traffic congestion. While the VRP assumes constant travel times, the TDVRP incorporates time-dependencies into their travel times, depending on the hour of the day when the travel starts. Many strategies with different levels of congestion avoidance have been applied in literature. Lower-level strategies include the modification of arcs or duration matrices and then solve the problem with algorithms for the regular VRP. Examples for those are lower-level strategies tested by Kok et al. (2012) or Lombard et al. (2018). Higher level strategies incorporate time-dependency directly into the algorithms. Exact algorithms as well as heuristics have been developed for this purpose. Both, lower and higher-level strategies improve the time-independent VRP, while higher-level strategies have been proven to result in more robust and better solutions in terms of the chosen KPIs. An overview of the (deterministic) literature is provided in Table 3.2. These strategies that were previously tested in literature will serve as the basis for the strategies that will be implemented and tested in this research.

Table 3.2: TDVRP Literature Summary

Strategy/ Algorithm	*Type*	*Travel Time Computation*	*applied by*
Nearest-Neighbor Heuristics	Heuristic	Travel Time Step Function	Malandraki & Daskin (1992)
Branch-and-Cut	Exact Algorithm	Travel Time Step Function	Malandraki & Daskin (1992)
Genetic Algorithm	Heuristic	Continuous Travel Time Function	Jung & Haghani (2001)
Parallel-Tabu Search	Heuristic	Speed Step Function	Ichoua et al. (2003)
Multi-ant colony system	Heuristic	Speed Step Function	Donati et al. (2008)
Tabu-Search	Heuristic	Continuous Travel Time Function	Osvald & Stirn (2008)
Ant Colony algorithm + insertion heuristics	Heuristic	Continuous Travel Time Function	Balseiro et al. (2011)
Dynamic Progr. + Average Travel Speeds	Exact Algorithm	Speed Step Function	Gromicho et al. (2012), Kok et al. (2012)
Dynamic Progr. + time-dependent VRP	Exact Algorithm	Speed Step Function	Gromicho et al. (2012), Kok et al. (2012)
Dynamic Progr. + time-dependent Shortest Path and VRP	Exact Algorithm	Speed Step Function	Kok et al. (2012)
Branch-and-Price	Exact Algorithm	Speed Step Function	Dabia et al. (2013)
GRASP + Average Travel Times	Heuristic	Continuous Travel Time Function	Lombard et al. (2018)
GRASP + time-dependent travel times	Heuristic	Continuous Travel Time Function	Lombard et al. (2018)

CHAPTER
FOUR

METHODOLOGY

In this chapter, the following research questions are answered: *'Which research framework can be set up to test different solving strategies for the VRP that incorporate historic traffic data?'*, *'Which data can be used to model realistic, time-dependent VRP instances?'*, *'Which strategies to incorporate historic traffic data should be tested?'* and *'Which KPIs could be used in order to compare the different strategies?'*. The framework of the research is set up and explained in order to give an overview of the design of the study (Section 4.1). The framework is composed out of three levels: The input, model and output level. Before these three levels are further explained, the specifications and assumptions of the VRP that is solved are presented (Section 4.2). Afterwards the three levels of the research framework are explained in detail. Firstly, the input to the model, namely the data that is used is presented (Section 4.3). This includes the presentation of geographical location data as well as the retrieval of travel times and the interpolation of travel times. That is, how we attain a continuous travel time function from discrete travel times in order to test the problem solution with time-dependent travel times. Secondly, the model itself is presented by introducing the strategies that are tested and the algorithm that is used to evaluate routing plannings (Section 4.4). Thirdly, the output of the model, namely the KPIs that are looked at to compare the tested strategies, are presented (Section 4.5).

4.1 Research Framework

In order to test different solving strategies for the VRP that incorporate historic traffic data, we set up the following framework, presented in Figure 4.1.
The framework is split up in three levels: The input, model and output level. On the input level, we start with defining a problem instance. This includes defining the location of a depot and multiple customers, the demand and time-window of each customer and the capacity of the vehicles. Afterwards, the (time-dependent) travel-time between each location is retrieved. This data is then processed in such a way that it serves as input for a strategy that will be tested. The model level includes two main parts. Firstly, the problem instance will be solved by the respective strategy. Out of the five strategies that will be evaluated, four do not consider time-dependencies while one of the five does. This will result in a routing solution. Secondly, the routing solution will be tested with time-dependent travel times. The model then outputs the resulting KPI realizations.

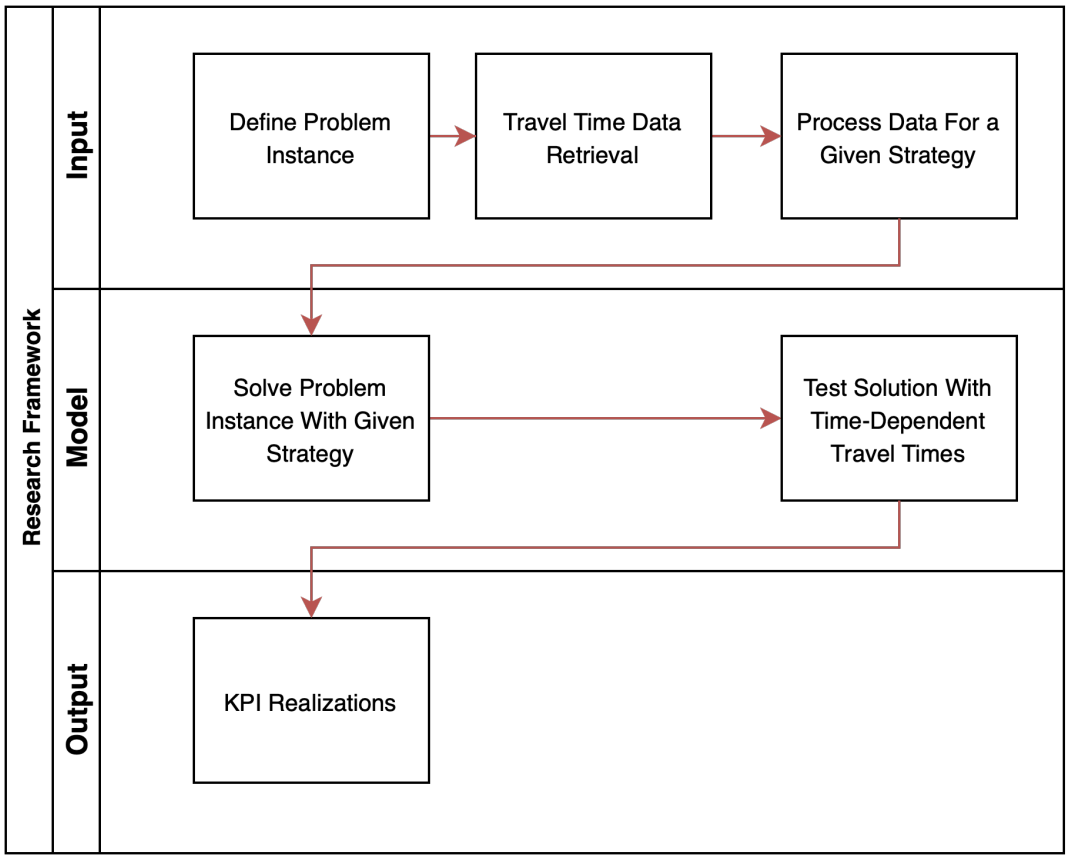


Figure 4.1: Research Framework

4.2 Problem Specifications & Assumptions

As for the general VRP, a set of vehicles needs to serve a set of customers. Additional constraints and assumptions apply. More specifically the problem has the form of a Time Dependent and Capacitated Vehicle Routing Problem with Time Windows (TDCVRPTW). We will now present the specifications and assumptions that constrain the VRP. This is done by splitting those specifications and assumptions into three parts: The general form, objective and constraints.

General Form

- Single-Day Optimization & Offline-Planning
- Single Depot

In our case, the set of customers needs to be served on one single-day. We are dealing with offline-planning because the optimization takes place before the trip starts and is not modified afterwards. There is only one depot from which each vehicle departs and to which all vehicles need to return.

Objective & Costs

- Objective is the minimization of the total driving time of all vehicles plus the total waiting time (in case a vehicle arrives early) of all vehicles. The number of vehicles is disregarded.

- Waiting at customer site is free (only costs time).
- No service times at customers' site.

The objective is the minimization of the total driving time by all vehicles plus the total waiting time by all vehicles. This time is further referred to as 'total time'. The waiting time is the time a vehicle has to wait in case it arrives early at a customer. Waiting is therefore allowed and only comes with time costs. Even though service times are relevant for most VRP cases, different service time settings are not expected to contribute to the answers to the research questions and are therefore disregarded. More specifically, we expect that the conclusions will be the same as the general problem of traffic congestion is also valid if service times are included.

Constraints

- Vehicles are capacitated. Same capacity for all vehicles.
- Unlimited number of vehicles.
- Each customer can only be served once. Each customer must be served.
- Customers have certain delivery hours (time-windows). One hard time-window per customer.
- Time-dependent travel times for all routes.
- No infeasible solutions allowed.
- Maximum route duration for one vehicle is restricted to 10 hours.

The problem is capacitated because each customer has a certain demand while the vehicles have a maximum capacity. Each vehicle has the same capacity. To assure that solutions are feasible, there is an unlimited number of vehicles available. Theoretically, each customer could be served by one customer specific vehicle which will however likely result in high travel times and therefore the number of vehicles will indirectly be minimized by the objective function. Each customer can only be served by one vehicle and all demand must be satisfied. Respectively, customers' demand must be lower or equal to the vehicle's capacity. A vehicle can only arrive at a customer during a given time-window as every customer has certain delivery hours. Each customer has one time-window per day. These time-windows are hard constraints, a vehicle must therefore arrive in or before that time-window to serve the customer. The travel times for each route change over the day. The problem is therefore time-dependent. All constraints are hard constraints. This means that they must be met and respectively, no infeasible solutions are allowed as the final solution to a problem. The European Agreement concerning the work of crews of vehicles engaged in international Road Transport (AETR) states that commercial vehicle drivers are allowed to work a maximum amount of 90 hours in two weeks and a maximum of 10 hours per day (Bundesverband Güterkraftverkehr Logistik und Entsorgung BGL e.V., 2020). We therefore choose the route duration limit for a vehicle to be 10 hours per day, where the route duration is defined as the driving time plus the waiting time. The route duration is therefore equal to the time the vehicle arrives back at the depot minus the time the vehicle left the depot, which is likely to be similar to the working time of the vehicle driver.

4.3 Input Data

This section addresses the collection of data that is used as input to the research. The goal is to transmit the reader an understanding of which internal data is used and what kind of external

data is retrieved and how it is retrieved. We therefore first look at internal data, namely pre-created geographical locations and then look at the external data namely travel times. Lastly, the interpolation of travel times is explained, that is, how we attain a continuous travel time function from discrete travel times in order to test the problem solution with time-dependent travel times.

4.3.1 Location Data

For testing purposes, the WDL previously created anonymized data of 250 locations. These positions are similar to previous cases and are based upon the population density of Germany. Therefore, more locations are based in regions with higher populations. It is not differentiated between depot and customer positions. This means that both, customer and depot locations, will be chosen from the same 250 locations. Figure 4.2 shows the respective plot with all locations marked with a (red) dot. From this plot we can see that many locations are positioned in regions such as the greater Berlin, Hamburg, Frankfurt, Ruhr and Munich area. The longest straight-line distance between two points is almost 793 km long (Flensburg and Rosenheim). The shortest distance between two points on the other hand measures only around 1 km (both locations in Bonn). From these 250 locations, the problem instances will be derived.

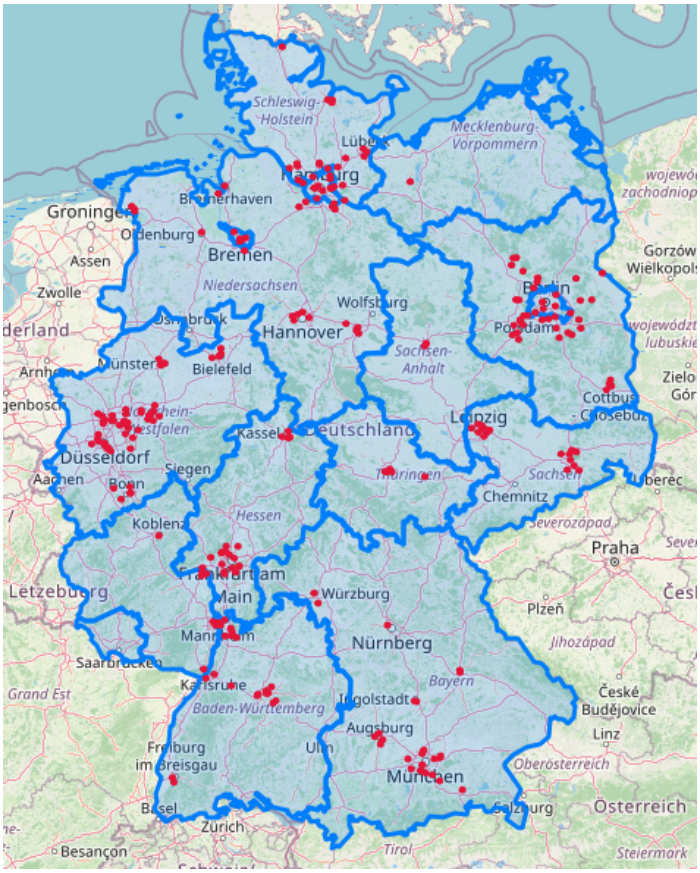


Figure 4.2: White-Labeled, 250 Locations in Germany (Source: WDL)

4.3.2 Travel Time Data

For the travel-time, two different data sources are used. The first one is the API of 'Openrouteservice' (*OpenRouteService Matrix*, 2020) which is currently used by the WDL. This API does not allow for

specifying the departure time nor allows for specifying the day of the travel but just gives average time-independent travel times. The second data source is the Bing Maps API (*Bing Distance Matrix API*, 2020). This API allows for specifying the departure time and gives time-dependent travel times respectively. Bing uses different sources to attain these time-dependent travel times. The highest-volume source of data for this purpose is GPS data from mobile phones Woodard et al. (2017). The reason for choosing Bing over other sources such as Google's Distance Matrix API (*Google Distance Matrix API*, 2020) is that it provides more generous free of charge usage options. The following two figures are a visualization of an example problem instance with three locations: A, B and C and three departure times t, $t+1$ and $t+2$. While Figure 4.3 shows just one time-independent travel time matrix as it would be returned by the 'Openrouteservice' API, Figure 4.4 shows a time-dependent three-dimensional traveltime matrix with three time units for each departure time. Neither matrix needs to be symmetrical.

$$\begin{bmatrix} 0 & c_{AB} & c_{AC} \\ c_{BA} & 0 & c_{BC} \\ c_{CA} & c_{CB} & 0 \end{bmatrix}$$

Figure 4.3: Time-independent 3x3 Travel-Time Matrix

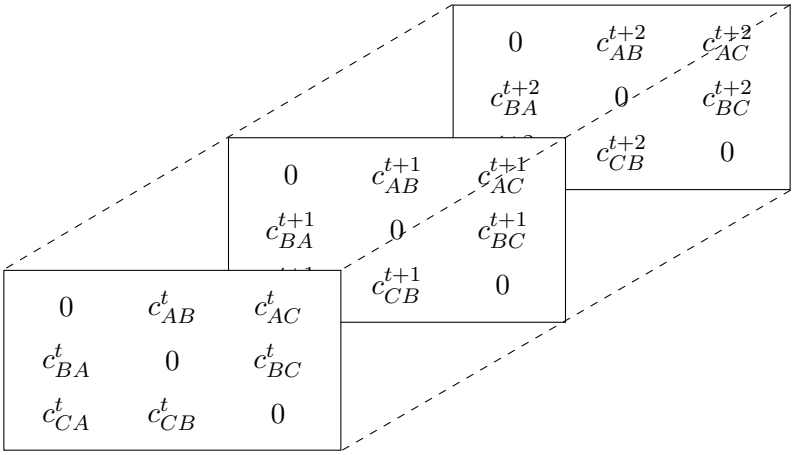


Figure 4.4: Time-dependent 3x3x3 Travel-Time Matrix

4.3.3 Interpolation of Travel Times

Previously we have seen that time-dependent travel time data between two locations is retrieved in a discrete way. This means that the travel time is not returned as a function over time but as multiple values for each specified departure time. As this is not a realistic setting, we need to apply some form of interpolation in order to transform the discrete data into a continuous travel time function. Like this, the solutions can be tested in a more realistic way.
Technically, the travel time on a certain day could be returned for every departure time with a frequency of a minute (e.g. 18:00, 18:01, 18:02...). However, for efficiency reasons, we retrieve travel

time data with a frequency of an hour in this research. More precisely, for a given day, we retrieve data starting at 3am and ending at 9pm. For each pair of location and day, we therefore get 19 travel times. Figure 4.5 and 4.6 show the travel time data over a given day for one pair of locations. The points (blue) represent the data as returned by the Bing API (one point per hour). The continuous straight line (yellow) represents the data which is returned by the Openrouteservice API: the travel time is constant over time.
In order to attain the travel time between two locations for each possible departure time between 3am and 9pm, we need to derive a continuous travel time function, that is, we need to interpolate between travel time data. Such an interpolation can be done by a piece-wise function, called *spline*. While Figure 4.5 shows a linear spline, Figure 4.6 shows a cubic spline (piece-wise polynomial function of 3rd degree). Choosing one of the splines depends on what we consider a more 'realistic' function. From looking at the figures we can see that the cubic spline is much smoother. Contrary, the derivative of the linear spline at the interior knot points (i.e. all points except the start and end point) is not defined, making the function changing its direction abruptly. According to Mancini (2014), "linear functions may be acceptable to represent sharp peaks but not wider ones. Polynomials, indeed, are able to better describe each type of peak." In order to make a good trade-off between the ability to describe sharp peaks and not overfitting on the data, cubic splines are chosen to derive a continuous travel time function given discrete travel time data. Next to that, we must keep in mind that the derivative may not be too small, in our case not lower than -1. In fact, this could lead to the scenario that departing at a later time will result in an earlier arrival, violating the FIFO condition. The reason why in our case this value is -1 is that both axis have the same scale, namely minutes. Assuming that we leave at time x and the traveltime is y minutes, then we would arrive at time $x + y$. If we now leave one minute later, at $x + 1$ and the travel time would also decrease to $y - 1$ then we would still arrive at the same time ($x + 1 + y - 1 = x + y$). If the travel time would however drop by more than one minute, we would arrive earlier when departing at $x + 1$ then when departing at x.

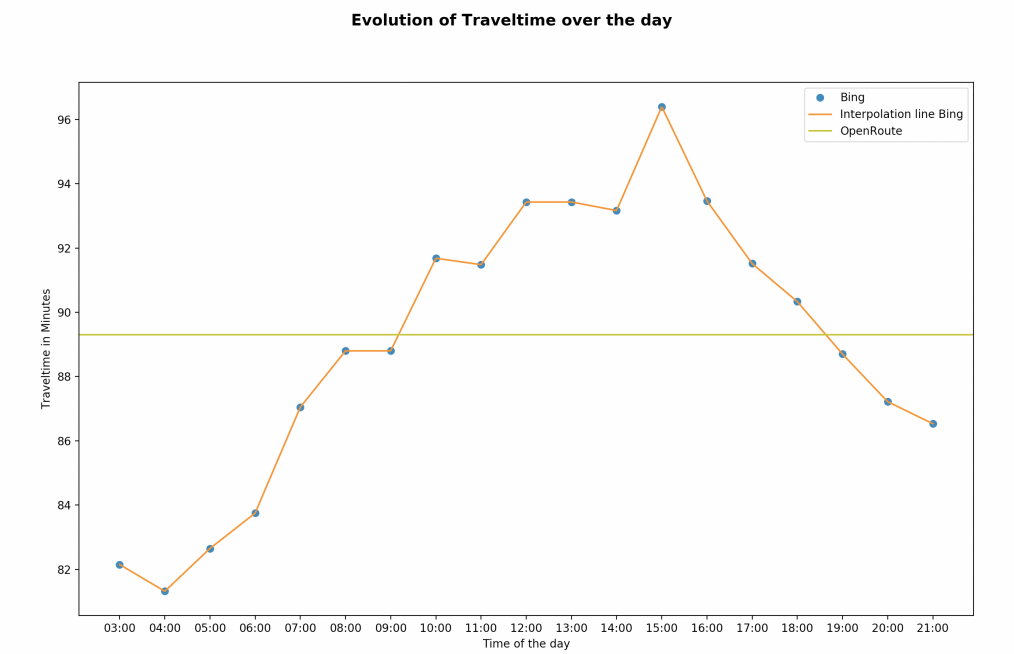


Figure 4.5: Interpolation of traveltime, Linear

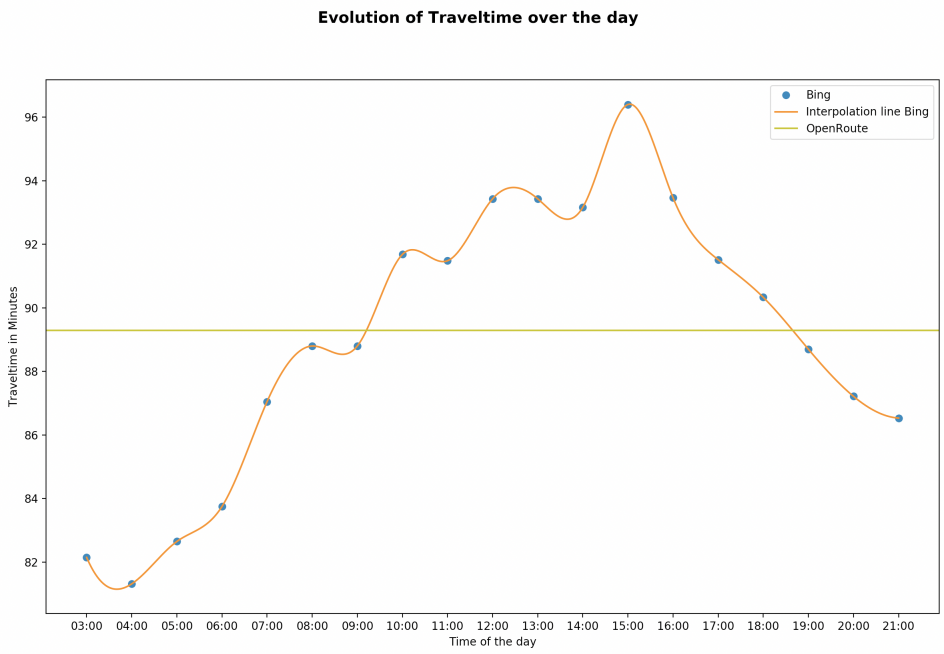


Figure 4.6: Interpolation of traveltime, Cubic

4.4 Model

The model consists of two main parts: Firstly a given problem instance is solved with a certain strategy and secondly the solution to this problem is tested with time-dependent travel times. We therefore first present the strategies that are tested and then the evaluation algorithm of how a solution is evaluated in a concrete manner. Out of the five strategies that are evaluated, four do not consider time-dependencies while the fifth strategy does so.

4.4.1 Strategies

In this research, five different strategies are tested. The strategies can be divided into two categories: In the first four strategies the problem will be formulated and solved as a Capacitated Vehicle Routing Problem with Time Windows (CVRPTW). Therefore, these strategies neglect time-dependencies but rather solve the problem with a 2-dimensional (asymmetrical) distance matrix as input (see Figure 4.3). The way these four strategies differ is that different distance matrices/ and or time-window inputs will be supplied to the algorithm. These four strategies are further referred to as the 'I-strategies', because the same algorithm is applied and only the input differs. Therefore, the 'I' stands for input. The CVRPTW will then be solved with a Tabu-Search Heuristic from the Google OR-Tools library (Perron & Furnon, 2019) in Python (see code in Appendix A.1). This algorithm from Google OR-Tools is also currently used by the WDL and serves as benchmark in many other scientific publications. The Tabu-Search heuristic of Perron & Furnon (2019) works as following: There are two lists. One list includes features of the solution that are not allowed (tabu) for following solutions, and one list that includes features that must be included in following solutions. Next to that, there are two phases. In the first phase which is called the diversification phase, it is tried to find solutions with different features than previously obtained. In the second phase which is called the intensification phase, it is tried to find solutions with similar (or the same) features to a given solution. The Tabu Search oscillates between these two phases: diversification to avoid being trapped in local optima and

also explore other parts of the search space and intensification to search more in details a promising neighborhood.

The fifth strategy incorporates time-dependent travel times by solving the TDCVRPTW with a Simulated Annealing algorithm with an objective value tabu-list. This strategy is further referred to as 'A-strategy' or simply 'Simulated Annealing', because it does not only differ by the input to the algorithm, but also by the algorithm itself. Therefore, the 'A' refers to algorithm. Next, each strategy is explained in detail. Table 4.1 provides an overview of the strategies and their characteristics.

Table 4.1: Solution Strategies

Strategy (type in brackets)	*Data Requirements*	*Level of Congestion Avoidance*
OpenRoute Data (I)	Time-indep traveltimes	Low
Compressing Time-Windows (I)	Time-indep traveltimes	Low
Bing Average (I)	Time-dep traveltimes	Medium
Worst Case Bing Data (I)	Time-dep traveltimes	Medium
Simulated Annealing with Tabu-List (A)	Time-dep traveltimes	High

Strategy 1: OpenRoute Data (I-Strategy)

This strategy corresponds to the current procedure of the WDL for routing projects that involve solving the CVRPTW. The travel time matrix is retrieved from the 'OpenRoute' API. This matrix is then supplied to the Google OR-Tools algorithm (Perron & Furnon, 2019). The data requirements for this strategy are therefore only a two-dimensional travel time matrix which does not need to be symmetrical. This strategy does neither avoid traffic congestion nor does it account for traffic congestion as it does not include any time dependencies. Its level of congestion avoidance is very low. It therefore serves as base-strategy for which possible improvements should be found.

Strategy 2: Compressing Time-Windows (I-Strategy)

This strategy is similar to the first strategy with the difference that time-windows at customers, that are used as an input to the OR-tools algorithm (Perron & Furnon, 2019), will be compressed. More precisely, the closing time at each customer will be subtracted by 10% of the time the customer is open. If the actual time open of a customer is for example 2 hours between 10am and 12am, then the time-window passed to the algorithm will be 10am until 11:48am (12am - 120 minutes * 0.1). It is expected that by doing so, delays will be minimized. The reason why 10% is chosen is that we could see from the data that travel times retrieved from Bing do not often vary more than 10% to the travel times from the 'OpenRoute' API. The data requirement is only a two-dimensional travel time matrix as it is the case for the first strategy. The strategy does not avoid any traffic congestion, it is however expected to result in more robust routes by avoiding delays.

Strategy 3: Bing Daily Average (I-Strategy)

In this strategy, the time-dependent travel time matrix (as retrieved from Bing) for the relevant day will be averaged along the third (time) dimension. This will lead to a two-dimensional travel time matrix which contains the daily average of all directed location pairs. Consequently, the matrix does not need to be symmetrical. This matrix then serves as input to the Google OR-tools algorithm (Perron & Furnon, 2019). As we are taking daily averages, the strategy requires the retrieval of time-dependent travel times. Similar to the first two strategies, this strategy does not account for traffic congestions and neither avoids traffic congestions because only an average is taken. It does however incorporate daily changes of travel times. Hence, the level of congestion avoidance is above the level of the first two strategies and classified as medium.

Strategy 4: Worst-Case Bing Data (I-Strategy)

Similar to the third strategy, also this strategy forms the time-dependent travel time matrix of Bing to a two-dimensional travel time matrix. For each directed location pair, the longest travel time (worst-case) for the specific day will be determined, leading to a two-dimensional (most likely) non-symmetrical travel time matrix. Again, this matrix then serves as input to the Google OR-tools algorithm (Perron & Furnon, 2019). In terms of data requirements, time-dependent travel times are needed. The reason why the longest travel time is taken is that by doing so, late arrivals can be almost totally eliminated as actual travel times will likely not be higher than the projected ones. In order to attain a feasible solution, the algorithm might therefore employ more vehicles or choose longer routes which will both come at higher costs in terms of total travel times. As late arrivals are avoided by accepting higher total times, the level of traffic congestion avoidance for this strategy is classified as 'medium'.

Strategy 5: Simulated Annealing with Objective Value Tabu-List (A-Strategy)

This fifth and last strategy involves the implementation of an algorithm that takes time-dependent travel times directly into account. We therefore suggest a Simulated Annealing algorithm with an objective value tabu-list. The reason why this is chosen is that it combines two powerful algorithms, namely Simulated Annealing and Tabu-Search. While the Simulated Annealing part makes sure that local minima can be escaped, the tabu-list is introduced to prevent cycling between solutions. The solutions attained during the algorithm are evaluated with the evaluation algorithm which will be presented in the next section. The constructive heuristic to initiate a first feasible solution is the nearest-neighbor heuristic. The heuristic is adapted for time-windows and the maximum route length constraint. The closest customer is determined by the average traveltime over the given day. The average traveltime is chosen because it also represents the expected travel time for a given day.

Algorithm 1: Nearest-Neighbor Heuristic

```
Input: Time-Windows; BingMatrix;
Output: Routelist, VehicleDepartTimes
while Not all customer have been delivered do
    Initialize New Vehicle i and its route: Routelist[i] ;
    Choose customer that opens earliest and was not delivered yet ;
    Add customer to routelist[i] ;
    while customer can be added to vehicle i do
        nextCustomer = Find closest customer from last customer currently in Routelist[i] ;
        if adding nextCustomer to Routelist[i] does not violate any constraints then
            Add nextCustomer to Routelist[i];
            mark nextCustomer as delivered;
        end
        else
            mark nextCustomer as not possible for current vehicle;
        end
    end
end
Determine VehicleDepartTimes ;
```

The departure time is determined by a backward recursive algorithm whose pseudo-code is depicted below. The algorithm is adapted from Savelsbergh (1992). The departure time at node P is the minimum of rather the closing time of node P or the departure time at node P+1 minus the travel time from P to $P+1$. The travel time is in this case the worst case travel time, as this prevents possible delays. The algorithm does likely not result in the shortest total time but it guarantees no delays at customers and is fast in terms of the algorithm run time. The pseudo-code is given below:

Algorithm 2: Vehicle Depart Time Heuristic

```
Input: BingMatrix; Routelist;
Output: VehicleDepartTimes
for each vehicle i do
    LeavingTime = closing time of last customer at vehicle i;
    for every Node P in vehicle, decreasing do
        LeavingTime at Node P = min(closing time P, LeavingTime at P+1 - driving time
         from P to P+1);
    end
    save LeavingTime at Node 0 to VehicleDepartTimes list
end
return VehicleDepartTimes ;
```

After a first feasible solution is generated with the constructive heuristic, the actual improvement algorithm is performed. As mentioned previously, the basis of the algorithm is a Simulated Annealing

algorithm. In order to prevent the algorithm from cycling between solutions, an objective value tabu-list is added. This tabu-list stores the objective value of the previous n solutions. It is then forbidden to go to solutions which rather violate the hard-constraints such as time-windows and the maximal route length, or whose objective value is on the tabu-list. The advantage of this kind of tabu-list is that no entire solutions have to be stored. As Malek et al. (1990) propose a tabu-list size of 7 to N for the TSP (where N is the number of cities), a tabu-list size of 10 is chosen. The pseudo-code of the Simulated Annealing Algorithm with Objective value tabu-list is shown below:

Algorithm 3: Simulated Annealing with Tabu-List

```
Input: InitialSolution ;
Output: BestSolution
temperature = 1;
CoolIteration = 0;
Set CoolingFactor;
CoolIteration = 0;
while CoolIteration < MaxIterations do
    CoolIteration = CoolIteration + 1;
    NewSolution = Choose Neighbor Solution;
    Evaluate Neighbor Solution;
    if NewSolution violates No constraints then
        if Newobjectivevalue < Oldobjectivevalue then
            currentSolution = NewSolution;
            if Newobjectivevalue < BestObjectivevalue then
                bestSolution = NewSolution;
            end
            add Solution to Tabu-List;
        end
        else
            if Temperature > RandomNumberbetween0and1 then
                currentSolution = NewSolution;
                add Solution to Tabu-List;
            end
        end
    end
    temperature = temperature * CoolingFactor;
end
```

Five different neighborhood structures are defined. These neighborhoods are chosen because they have been applied by Wang et al. (2019) in a local search algorithm, resulting in state-of-the-art results for the cumulative capacitated vehicle routing problem. In the node insertion neighborhood, a node from one route is inserted into rather another place of the same route or into a different route. In the case below, node 4 is inserted into the first route.

1. Node Insertion

0-1-2-3-0 and 0-4-5-6-0 → 0-1-2-3-4-0 and 0-5-6-0

In the node-node exchange neighborhood, two nodes from rather the same or different routes are swapped.

2. Node-Node Exchange

0-1-2-3-0 and 0-4-5-6-0 → 0-4-2-3-0 and 0-1-5-6-0

The arc insertion neighborhood involves the insertion of two consecutive nodes (defined as an 'arc') to a different or the same route. In the example below, the arc '4-5' is inserted to the end of the first route.

3. Arc Insertion

0-1-2-3-0 and 0-4-5-6-0 → 0-1-2-3-4-5-0 and 0-6-0

As the name suggests, the arc-arc exchange swaps two arcs (e.g. arc '1-2' with arc '4-5').

4. Arc-Arc Exchange

0-1-2-3-0 and 0-4-5-6-0 → 0-4-5-3-0 and 0-1-2-6-0

The last neighborhood structure is the arc-node exchange which involves the exchange of a single node with an arc. As for the other neighborhood structures, the arc and node to be swapped do not necessarily need to be part of different routes. In the example below, arc '4-5' is swapped with node '1'.

5. Arc-Node Exchange

0-1-2-3-0 and 0-4-5-6-0 → 0-4-5-2-3-0 and 0-1-6-0

In every improvement heuristic it is important that the solution space is connected through the neighborhood structure so that every (and preferably better) solution can be reached by the current solution. We therefore show now that with our chosen neighborhood structure, all solution spaces are connected. Through the node insertion heuristic, the state where all customers are in different routes can be reached without the violation of any constraints (time-windows, truck capacity and maximal route length). In the worst case, we can assume a route that contains all customers and each customer will be removed one by one until there is only one customer left in the initial route. The state can therefore be reached in a maximum of $n-1$ steps, where n is the number of customers. From this state, all other feasible solutions are reachable by inserting the customers one by one into their new position. Again, no constraints are violated when directly going from the solution where all customers are in different routes to another feasible solution. As for the destruction of the solution, in the worst case, each customer would need to be repositioned to the same route which will then take at most $n-1$ steps. One customer does not need to change its position in the route list as the other customers can be ordered around it. Therefore, all feasible solutions are connected through the node insertion neighborhood structure in at most $2n-2$ steps.
As test runs have shown that the arc-insertion, arc-arc exchange and arc-node exchange neighborhoods often result in infeasible solutions, we set the probabilities that these neighborhoods are chosen to 0.1 and the probabilities for the node insertion and node-node exchange neighborhood to 0.35. With these probabilities, it is more likely to attain a feasible solution then if probabilities would be equal for all neighborhood structures.
The cooling scheme takes the form: $t_k = t_{k-1} * a$, where k is the current iteration, $0 < a < 1$ and $t_1 = 1$. It has therefore similar properties to a negative exponential function. The cooling factor a is chosen depending on the maximum number of iterations.

4.4.2 Evaluation Algorithm

After a solution to a problem instance is attained by the just mentioned strategies, the next step is to evaluate these strategies with time-dependent travel times. We therefore set up an evaluation algorithm that takes four inputs: the time-windows of the problem instance, the time-dependent Bing matrix (see Figure 4.4), the route for each vehicle and when the vehicle should depart. The $RoutePerVehicle$ is a list of K sub-lists where K is the number of vehicles. It includes the sequence in which the customers are visited. In the below example there are 2 vehicles used and 4 customers to be served. Vehicle 1 departs at node 0 (the depot) and then serves customer 2 and 4 (in this exact sequence) before it returns to the depot. The same logic applies for the second vehicle.

$$RoutePerVehicle = [[0, 2, 4, 0], [0, 1, 3, 0]]$$

The $VehicleDepartTime$ determines at what time the vehicles should leave the depot (start the route). In the example below, vehicle 1 leaves the depot at time 0 and vehicle 2 at time 30.

$$VehicleDepartTime = [0, 30]$$

The algorithm outputs a three-dimensional list, which will be further referred to as 'Evaluation List'. The list includes K sub-lists where k is the number of vehicles (routes). Each of these K sub-lists includes $N + 2$ sub-sub-lists where N is the number of customers visited by this vehicle. The two sub-sub-lists added refer to the start and end of the route (the depot). The first element of the sub-sub-list includes the index of the node. The second element includes the arrival time of the vehicle and the third element the time the vehicle leaves. In the example below the first vehicle departs from the depot (index 0) at time 0. The vehicle then arrives at time 15 at customer 2 and leaves this customer at the same time to customer 4. The vehicle arrives at time 35 at customer 4. As the opening time for this fictional customer is at time 40, the vehicle can only leave customer 4 at that opening time. The vehicle then returns to the depot and arrives at time 60.

$$SimList = [[[0, 0, 0], [2, 15, 15], [4, 35, 40], [0, 60, 60]], [[0, 30, 30], [1, 35, 35], [3, 45, 45], [0, 55, 55]]]$$

The pseudo-code below shows how the evaluation list is attained exactly. For each route the following is done: We start and depart from the depot at the time specified in the VehicleDepartTime list. For each following node (each customer and the depot upon return) we then attain its index and also the index of its previous node. The travel time from the previous to the current node is then calculated with a 'GetTravelTime' function which calculates the travel time with the time-dependent Bing data matrix. The function is explained in detail in the 'Interpolation of Travel Times' section. The arrival time of the current node is then the time we left at the previous node plus the travel time. The departure time of the current node is rather the arrival time or in case of early arrivals, the opening time of the current node. The current node, its arrival and departure time are then added to the evaluation list. With this evaluation list it is then straightforward to calculate all relevant KPIs which are described later.

Algorithm 4: Evaluation Algorithm

```
Input: RoutePerVehicle; VehicleDepartTime; Data; BingMatrix;
Output: SimList
for i=0 to NumberOfVehicles do
    LeavingTime = VehicleDepartTime[i] ;
    SimList[i].append(0, LeavingTime, LeavingTime) ;
    for j=1 to length(RoutePerVehicle[i]) do
        From = RoutePerVehicle[i][j - 1];
        To = RoutePerVehicle[i][j];
        TravelTime = GetTravelTime(BingMatrix, From, To, LeavingTime);
        ArrivalTime = SimList[i][j-1][2] + TravelTime ;
        LeavingTime = max(Data['TW'][To]['OpeningTime'] ,ArrivalTime) ;
        SimList[i].append([To, ArrivalTime, LeavingTime]) ;
    end
end
```

Validation of the Evaluation Algorithm

Law (2005) states that the validation of (simulation) models is the most important part in constructing models. It is also crucial in this thesis to test the validity of the evaluation algorithm as the experiments and their implications are based on it. To test the the evaluation algorithm we are comparing its results to the results provided by the Google OR-tools algorithm (Perron & Furnon, 2019). As OR-tools returns the results for time-independent travel times we need to adjust our evaluation algorithm such that it also evaluates with time-independent travel times (the same travel times as provided to OR-tools). We then compare the results for a given route and departure time. For multiple scenarios and routes, the results returned by OR-tools and the ones returned by our evaluation algorithm are similar. We can therefore conclude that the evaluation algorithm is valid.

4.5 Key Performance Indicators

This section addresses the output of the model, namely the Key Performance Indicators (KPIs). While the objective function of the problem is to minimize the total time of the routes (travel time + waiting time), also other KPIs will be examined. We therefore consider the following eight KPIs:

- Travel Time
- Waiting Time
- Vehicles used
- Number of late arrivals at customers
- Lateness time at customers
- Number of late returns to depot
- Lateness time at the depot

- Number of Routes with Lateness

As the objective function of the problem instances is the minimization of the total driving time by all vehicles plus the total waiting time by all vehicles, the realized travel and waiting time are indispensable. As described previously, the waiting time is the time a vehicle has to wait in case it arrives early at a customer. Next to that, also the number of vehicles used will be looked at. The following five KPIs address the robustness of the solution in terms of time-window violations. Firstly, the number of late arrivals at customers will be examined. Next to that, also the total lateness time at customers will be looked at. Secondly, analogous to the previous KPIs, also the number of late arrivals and the total lateness time at the depot will be looked at. Lastly, in order to check if single routes cause most late arrivals or if late arrivals are more distributed over different routes, we will look at the number of routes with a lateness. While all these KPIs are relevant, the focus of the comparison will lay on the total time and the lateness time at customers.

4.6 Conclusion

The goal of the methodology chapter was to provide a detailed description of the methodology used in this study. This was done by answering several research questions. Firstly, it was asked 'which research framework can be set up to test different solving strategies for the VRP that incorporate historic traffic data?'. Therefore, a research framework with three levels was presented. An input, model and output level. For each of these three levels, a separate research question was answered. In order to provide input to the model, the following research question has been answered: 'Which data can be used to model realistic, time-dependent VRP instances?'. A problem instance with location data in Germany provided by the WDL is defined and travel time data is retrieved from the Bing API. After a problem instance has been defined and the relevant data has been retrieved, it is asked 'which strategies to incorporate historic traffic data should be tested?'. Five strategies of different levels have been suggested. In the first four strategies, only the input of the problem instance is changed while the algorithm is a tabu-search from the Google OR-tools (Perron & Furnon, 2019) which is currently used by the WDL. These four strategies are therefore referred to as 'I-strategies' where the 'I' stands for input. The first strategy involves solving the problem with time-independent data retrieved from the OpenRoute API. In the second strategy, time-windows are compressed to prevent delays while still time-independent data retrieved from the OpenRoute API is used. In the third strategy, travel times are averaged over a day by using time-dependent data from Bing. In the fourth strategy, for each directed location pair, the longest travel time (worst-case) for the specific day is determined. In the fifth and last strategy, the problem will be solved with a Simulated Annealing algorithm with an objective tabu-list which directly takes time-dependent travel times into account. This strategy is referred to as 'A-strategy' (where 'A' stands for algorithm) or simply 'Simulated Annealing'.
Before evaluating these strategies, relevant KPIs need to be defined which will be the basis of a comparison. For this reason, the following research question has been answered: 'which KPIs could be used in order to compare the different strategies?'. While many specific KPIs could be considered for a case in the future, the focus of this study will be on the total time of all routes and the lateness time at customers. While Chapter 5 includes an exploration and validation of the data, the research framework presented will be applied in Chapter 6 in order to compare the different strategies.

CHAPTER
FIVE

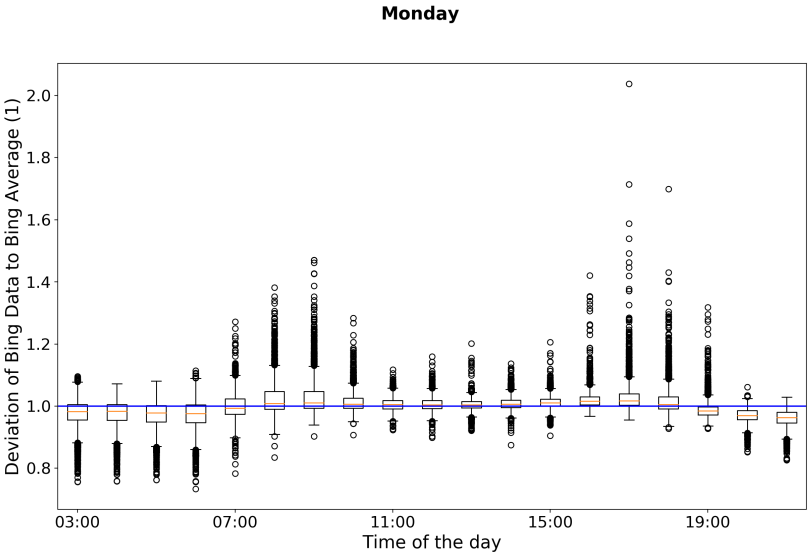
DATA EXPLORATION

The goal of this exploratory data analysis is to get an understanding of the data that is used by answering the following sub-research question: *'Which characteristics does the collected travel time data have and which influences do these characteristics have on the experimental design?'*. We define three scenarios with 30 customers and a maximum distance of 300 km from the depot to the customer points. The travel time data for the starting times between 3am and 9pm is then collected for each working day. These configurations are chosen because they present a realistic environment for routing problems commonly solved by the WDL. Next to that, for generalization purposes, the mode of transportation is cars. As for the Bing API we need to specify a day in the future, for each scenario we have chosen a different week in July 2020. This can come with limitations as it might be the case that July is very different from other months, e.g. due to vacations. Testing each working day in different months would however be out of the scope of this thesis and we assume the general findings to be valid for every month. We now first look if the traffic density distribution is different between week days (Section 5.1). Afterwards we compare the Bing data with travel time data from OpenRoute (Section 5.2). The section ends with the analysis of outliers, that is, routes with high travel time variance over a day (Section 5.3).

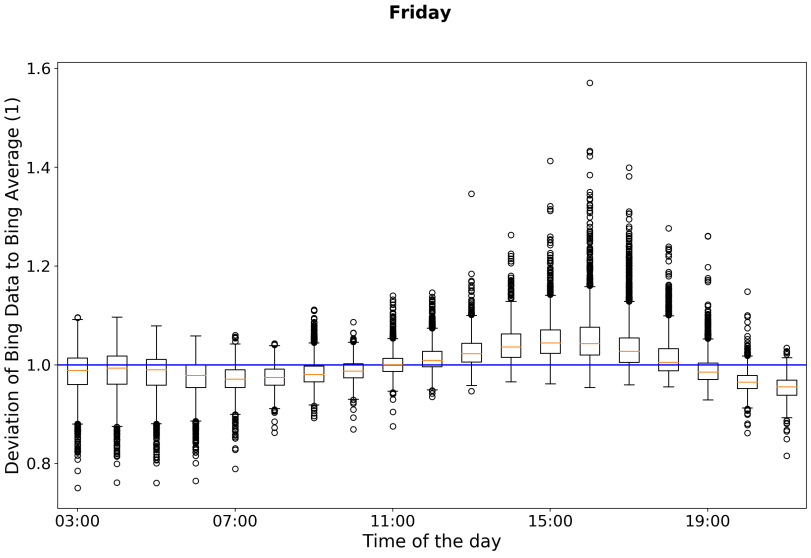
5.1 Week Day Variation

Figure 5.1 shows the traffic density variation for the day of Monday and Friday. The distributions for the other workdays are to be found in the appendix. The plots were created as follows: For each route, the average travel time over the day was computed. For each starting hour, the travel time is then divided by the average travel time for this route. For a given day, we therefore attain the variations of the travel time over the day. The straight (blue) line represents the average (100%).

For both days, in the early morning from 3am to 6am there are many outliers whose travel time is lower than the day average (up to 25% lower). Afterwards, we can see that the morning peak at around 8:00 is much higher for Monday, as many outliers do exist for which the travel time is more than 15% higher than the day average. On Monday, the travel time then decreases around noon. This characteristic is not visible on Friday. On both days in the afternoon, travel times increase again while this increase is much higher on Friday than on Monday. For that time, many routes vary by more than 30% to their mean value while one route on Monday at 5pm even varies by 100% compared to its Monday mean, so the travel time for this particular route is expected to be twice as high at Monday 5pm then its Monday average. The travel times in the evening then decrease again on Monday and on Friday. Table 5.1 presents a distance matrix for the different working days. The numbers represent the average percentage of the absolute difference between the data represented in Figure 5.1. For example,



(a)



(b)

Figure 5.1: **(a)** Traffic Density Variation - Monday **(b)** Traffic Density Variation - Friday

the travel time between city i and j on Monday at 3am differs by 5% to the Monday average. The travel time between those two cities on Friday at 3am differs by -3% to the Friday average. The absolute difference for the route at 3am between Monday and Friday is then 8%. Respectively, the same is done for all possible routes at every time of the day and the average of this difference is then taken.

Table 5.1: Distribution Comparison per Weekday

	Monday	*Tuesday*	*Wednesday*	*Thursday*	*Friday*
Monday	0	1.77	2.14	2.31	3.00
Tuesday	-	0	1.55	1.81	2.80
Wednesday	-	-	0	1.41	2.28
Thursday	-	-	-	0	2.15
Friday	-	-	-	-	0

We can see that Monday and Friday are most different (3%). Wednesday and Thursday have the most similar traffic distribution over the day (1.41%). These two observations can be confirmed with the plots in Figures 5.1, A.2 and A.3. Next to that, it is noticeable that the difference increases for each day from Monday to Friday. The distribution therefore changes gradually in the same direction from Monday to Friday.
We now look if the level of the travel time changes for different days of the week. We therefore compute the average travel time per working day and scenario. The results are presented in Table 5.2.

Table 5.2: Average Travel time in minutes- per Working Day and Scenario

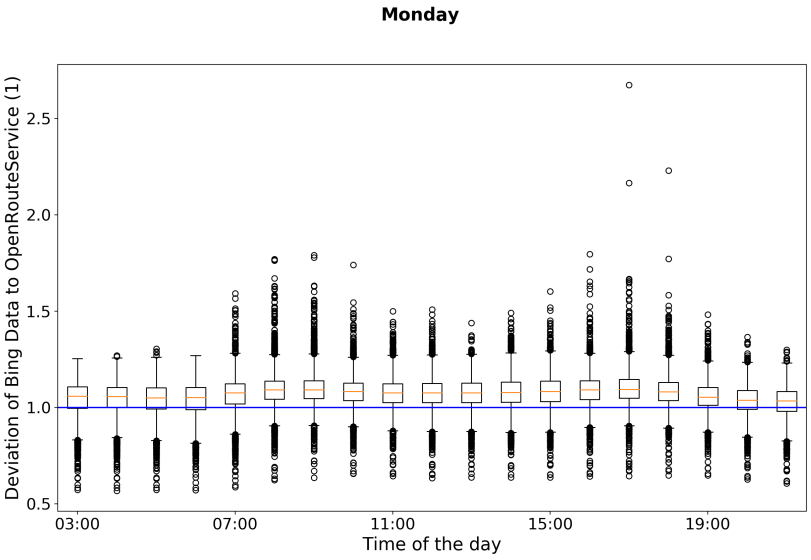
	Monday	*Tuesday*	*Wednesday*	*Thursday*	*Friday*
Scenario 1	143.73	148.52	149.15	146.32	144.61
Scenario 2	92.83	93.07	93.23	90.98	92.18
Scenario 3	172.33	175.95	176.99	173.66	174.63
Weekday average	136.30	139.18	139.79	136.99	137.14

In all scenarios, Tuesday and Wednesday are the days with the highest average travel times. While Thursday and Friday have similar travel times, travel times are shortest on Monday.

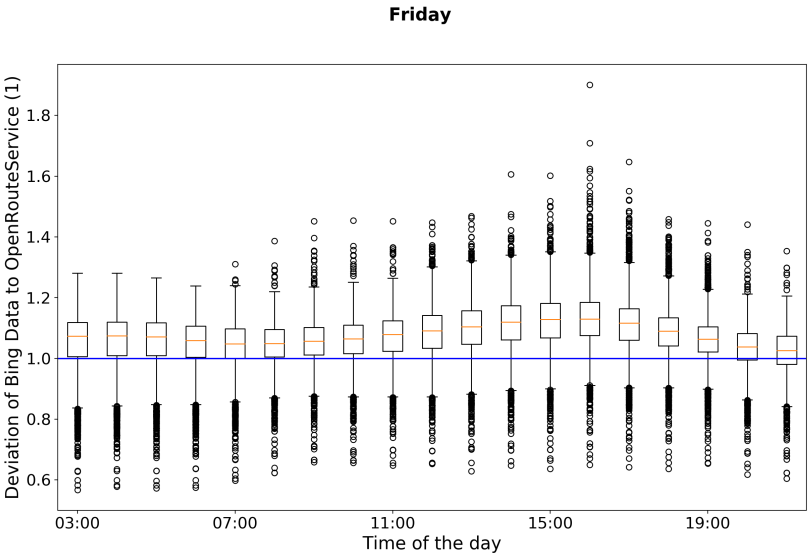
5.2 Data Validation with Openrouteservice

Currently the WDL is retrieving data from the Openrouteservice API which is a free data source. As already explained, the Openrouteservice API only allows for time-independent travel times between two locations. We therefore now look how the Bing data differs from the Openrouteservice data. By doing so, the data retrieved can also be validated if the data retrieved by the two sources resembles

each other. Figure 5.2 shows how the Bing data differs from the Openrouteservice data. The figure was attained by dividing for each route and starting time, the Bing travel time by the Openrouteservice travel time. The straight (blue) line represents the Openrouteservice basis.



(a)



(b)

Figure 5.2: Traffic Density Variation with Openrouteservice as base - Monday and Friday

We can see that for both, Monday and Friday, and all the starting times, the median deviation is above 1 and varies at around 1.1. This means that for most routes and times, Openrouteservice gives lower travel times than Bing. In fact, the travel times returned by Bing are on average 10% higher than the Openrouteservice travel times. Next to that, on both days there are many outliers whose deviation is above 50%, especially in peak hours in the morning and afternoon. A reason for that might be that Openrouteservice does not account for traffic but sets the speed of the arcs to

their maximum. This is an interesting finding as biased travel times can lead to non-robust routing solutions.

5.3 Routes with high variability

In the previous subsections we have seen that travel times are variant over the the day and this distribution is different for different days. Now, we look further at those highly variable routes as if not accounting for their variability, they might lead to non-robust routes. We take a look at two measures of the variability: the variance and the coefficient of variation. While the former one gives an absolute measure for the variability, the latter one standardizes the variability by the mean.

5.3.1 Measured by Variance

The following Figure 5.3 shows the 10 routes with the most variable route durations. For each route, only the day with the highest variance is kept and the other days for that route are discarded.

Figure 5.3: 10 routes with highest variance in route duration

Firstly, it is noticeable that many of these routes share common parts of their routes. In fact, all the routes pass the highway 'A1' from North Rhine-Westphalia to Bremen or Hamburg. The average distance of those points is around 468 km. The average travel time of those routes is 243 minutes, so 4 hours and 3 minutes. As the average route duration (average for all routes in every scenario and on every day) is 137 minutes, we can already deduct that longer routes likely have a higher variance. Besides that, it is remarkable that all 10 routes pass the state of North-Rhine Westphalia which is the most populated state of Germany. For each route on a given day, we therefore bin the routes

according to their average route length on that day and plot the variance distribution for each of these bins. The following boxplot (5.4) shows the results.

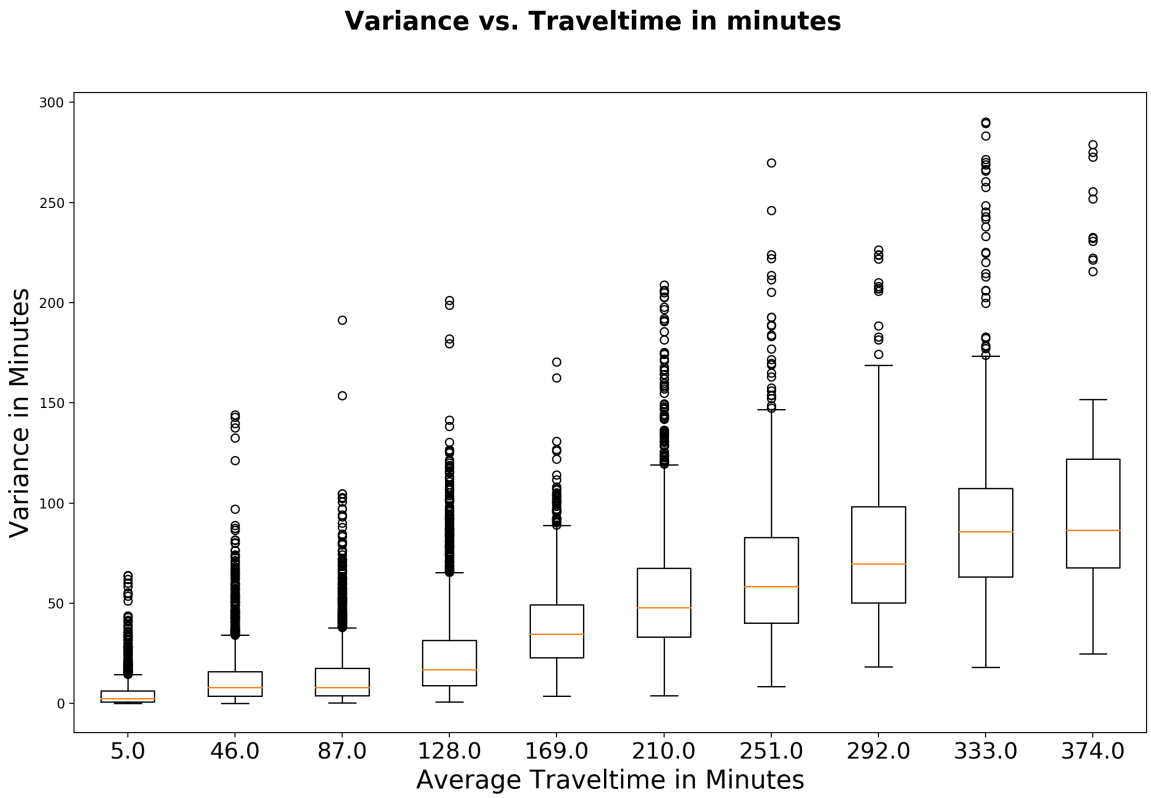


Figure 5.4: Variance in relation to route length

As it can be expected, we can see that long routes also have a high variation. This is an interesting observation as routes with a high variation might cause problems in routes. In future research, these routes could be treated differently in the routing algorithms.

5.3.2 Measured by the Coefficient of Variation

We now look at the Coefficient of Variation (CoV). The CoV is chosen to see the variance in the routes in relation to their mean. The CoV of the random variable X is defined as:

$$CoV(X) = \frac{\sqrt{Var(X)}}{E(X)}$$

Figure 5.5 shows the 10 routes with the highest CoV in the three scenarios (out of 2790 total routes). The CoV is computed for each route and day. For each route, only the day with the highest CoV is kept and the other days for that route are discarded.

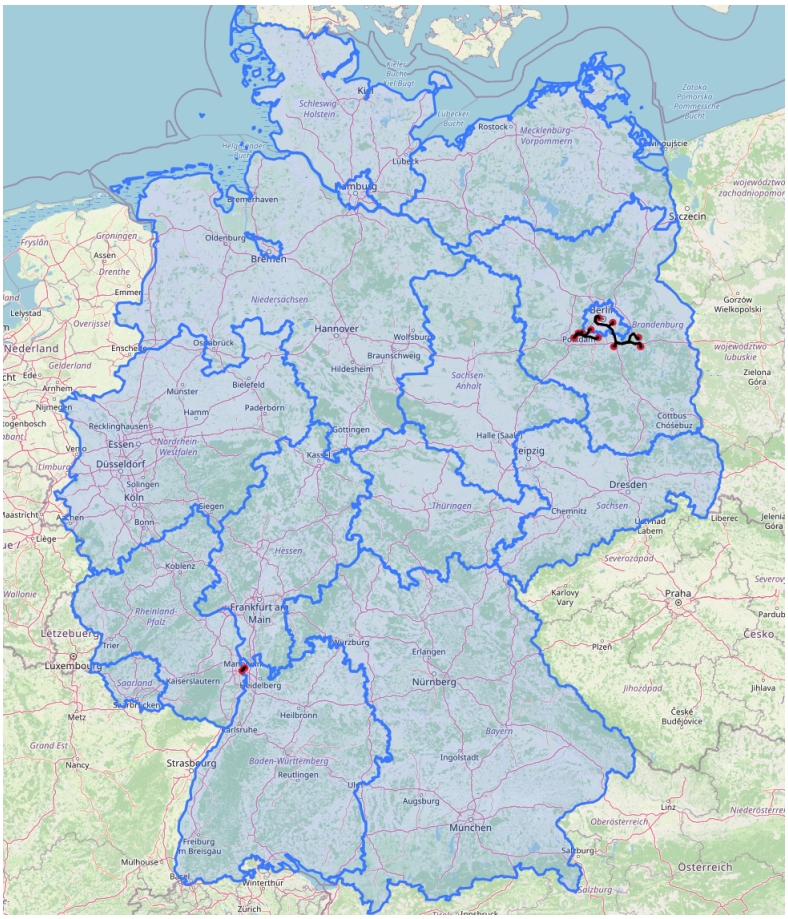


Figure 5.5: The 10 routes with highest coefficient of variation in route duration

8 out of the 10 routes are located in the greater area of Berlin. The two other routes are located in the city of Mannheim. The average route length for those 10 routes is around 26 km and the average duration is 29 minutes. As stated before, the average route duration is 137 minutes. Therefore it can be stated that shorter routes that pass densely populated areas are more prone to a higher CoV. We now take a closer look how the variation of travel times changes depending on the route duration (Figure 5.6).

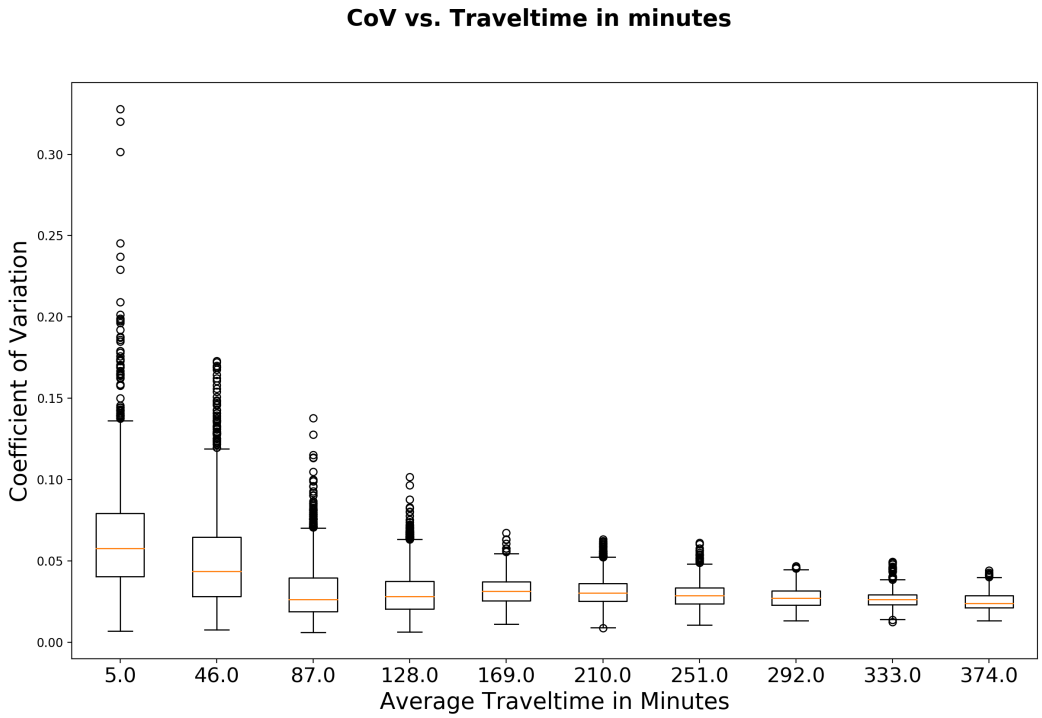


Figure 5.6: Coefficient of Variation in relation to route length

We can see that shorter routes have a higher CoV than longer routes. Especially short routes have much more outliers than longer routes.

5.4 Conclusion

In this chapter an exploratory data analysis was performed in order to answer the sub-research question: *'Which characteristics does the collected travel time data have and which influences do these characteristics have on the experimental design?'*. This was done by analyzing the travel time data of three scenarios with 30 customers each. For each of these scenarios, data for every working day was collected from 3am to 9pm on an hourly basis from the Bing Distance Matrix API and time-independent travel times from the OpenRoute API.
It is visible that the travel time distribution over the day is slightly different for every working day while Monday and Friday are most different. This is due to the fact that Monday has a strong morning and less strong afternoon peak while Friday only has a very strong afternoon and almost no morning peak. Next to that, it is noticeable that the difference increases for each day from Monday to Friday. The distribution therefore changes gradually in the same direction from Monday to Friday. It is therefore important to take different working days into account when designing the experiments. A bit surprisingly, we have seen that Wednesday and Tuesday have slightly higher travel times than the other days. When comparing the Bing data to the OpenRoute data we can see that on average the Bing data is around 10% higher than the OpenRoute travel times. There are however many outlying routes whose travel times are up to 150% higher or 50% lower than the corresponding travel time estimation by OpenRoute. Lastly we have seen that long routes are prone to a higher variance over the day while for short routes the variance over the day relative to the average route duration is higher. For the experimental design it is therefore important to include scenarios with different average route durations.

CHAPTER
SIX

EXPERIMENTS

The goal of the chapter is to answer the following sub-research questions: *'How can the collected data be used to model realistic VRP instances?'*, *'To what degree does predictable traffic intensity influence current routing plannings in terms of the chosen KPIs?'*, *'How do the tested strategies perform in terms of the chosen KPIs?'* and *'How can the results be validated?'*. We therefore first explain how the VRP instances are determined (Section 6.1). For each of the instances the following three steps are then performed: Firstly, we apply all five strategies described in Chapter 4 to solve the instance. Secondly we then collect the routes and departure times they generated and lastly we evaluate how these routes perform under time dependent travel times. The results of these evaluations are then analyzed by looking at the KPI realizations of the different strategies (Section 6.2). Lastly, a validation of the results with a second data-source for time dependent travel times is performed, namely with data from the 'Google Distance Matrix API' (Section 6.3).

6.1 Experimental Design

We define two main parameters for our problem instances: The number of customers and the maximum air distance between the depot and each customer. We choose three different settings for the number of customers: 10, 20 and 30 customers. These numbers are chosen as they present realistic case sizes and should show evidence for differences in case those exist between different problem sizes. Also we choose three different settings for the maximum air distance between the depot and each customer: 100km, 200km and 300km. Again, these number represent typical real-life distances. We therefore have 9 different instance settings. For each of these 9 instance settings, 4 instances are determined randomly from the locations presented in Section 3.1 of Chapter 4. We therefore attain 36 instances. The following Table (6.1) visualizes the structure of these different settings.

Table 6.1: Structure of Problem Instances

	Maximum Distance		
# of Customers	100km	200km	300km
10 Customers	Inst 01-04	Inst 05-08	Inst 09-12
20 Customers	Inst 13-16	Inst 17-20	Inst 21-24
30 Customers	Inst 25-28	Inst 29-32	Inst 33-36

As we have seen in Chapter 5 that travel time distributions differ per weekday, travel time matrices for a random Monday, Wednesday and Friday in July 2020 for all instances are retrieved. These three weekdays are chosen because we have seen that the traveltime distribution over a single days changes gradually from Monday until Friday. We therefore include the two most distinct days, Monday and Friday, and a third day in between, namely Wednesday. An example of an instance with 30 customers and a maximum distance between the depot and each customer of 200km is shown in Figure 6.1. The depot is marked with an icon.

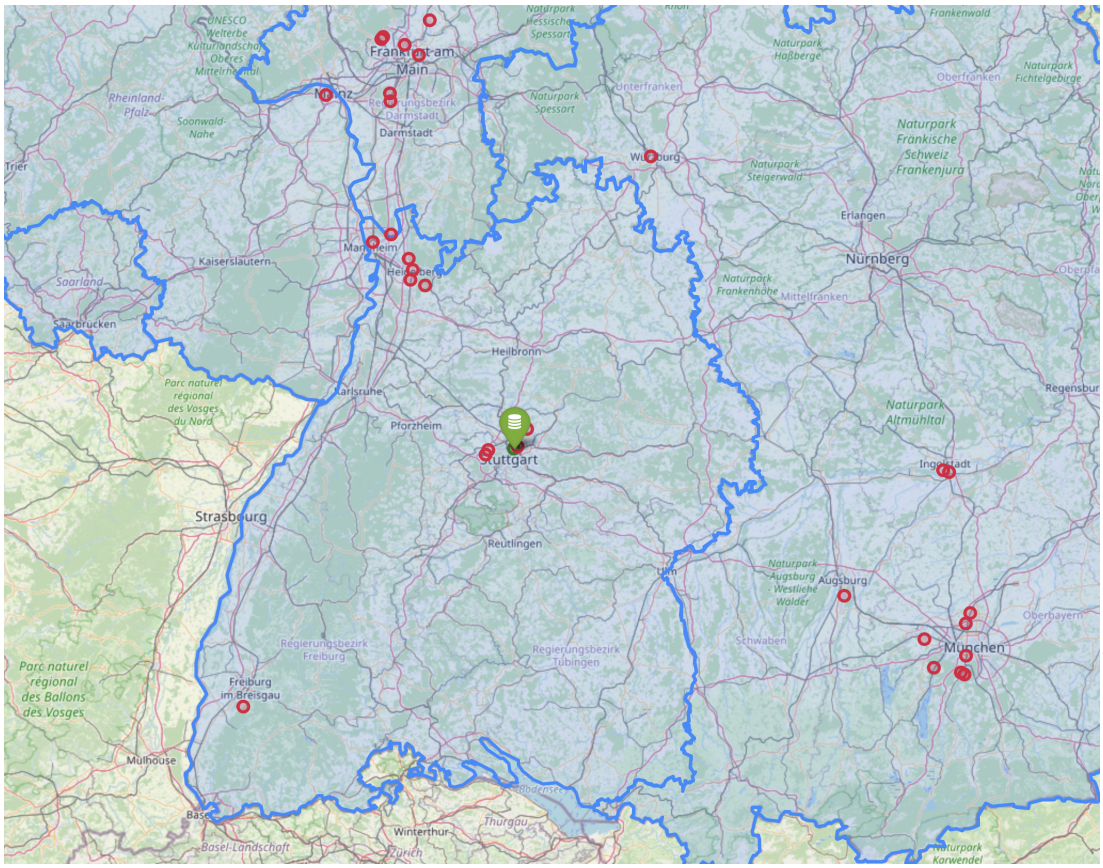


Figure 6.1: Example Instance with 30 customers and maximal 200km distance

After the travel time matrix of a problem instance is defined, each customer will get a time-window assigned. Also here, we evaluate three different time-window settings. In the first setting all customers are open for 0.5 hours. In the second setting for one hour and in the last setting all customers are open for 3 hours. In all settings, the earliest opening time is 6am and the latest closing time 6pm. Opposed to the time-windows, only one setting of the customers' demand is applied, namely a uniform distribution between 1 and 3. Each vehicles' capacity is set to 15. For each of these different scenario setting combinations, the demand and time-windows is determined randomly with 10 different random seeds. We therefore attain 36*3*3*10 = 3240 different scenarios which are solved with the five strategies. Even though we are aware that the problem complexity is likely to increase faster than any linear function, we choose the following algorithm run times: for cases with 10 customers it is set to 10 seconds, the run time for cases with 20 customers is set to 20 seconds and the run time for cases with 30 customers is set to 30 seconds. These times are chosen because they allow for several 1000 iterations and yet are feasible times to test as they do not consume too much time. A run time of 60 seconds would already require almost 4 days for only the cases with 30

customers, which is out of scope for this thesis. The experiments are performed on an Intel Core i5 Dual-Core with 3.1 GHz and 16 GB RAM.

6.2 Results

In this section the results of the experiments are presented. In order to attain a clear overview, results are split by problem size, i.e. the number of customers (Section 6.2.1), the maximum distance between customers to the depot (Section 6.2.2) and by time-window size (Section 6.2.3). In the following, when we speak of long/ short routes, we mean long/ short routes as measured by time, as this is also the KPI in the objective function.

6.2.1 Results by Problem Size

10 Customers

For the case with 10 customers, Figure 6.2 shows a boxplot that summarizes the results of the 1080 scenarios (as explained in the previous section). For each strategy, the left-sided (blue) boxplots represent the deviation of the total time to the shortest time found. A value of 5 would mean that the total time attained in a certain scenario is 5% longer than the shortest total time that was attained in the same scenario by another strategy. The right-sided (orange) boxplots represent the lateness at customers in minutes.

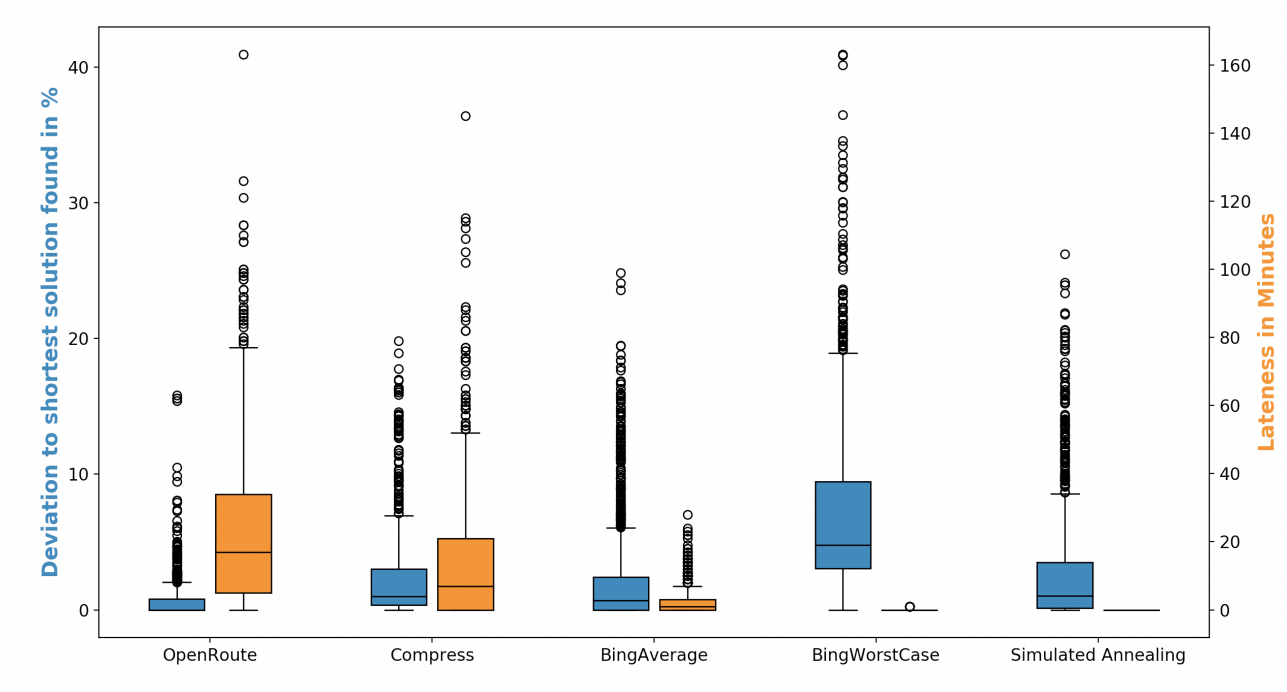


Figure 6.2: Deviation to shortest solution found and lateness in minutes in cases with 10 customers

It can be seen that the OpenRoute strategy resulted in the lowest total times, while the BingWorstCase strategy resulted in the highest total times. There are many outliers for both travel time and lateness for all five strategies. This is a sign that the performance of a routing is very dependent on the specific settings. On average, the OpenRoute strategy is only 0.7% above the shortest solution

found (by the five strategies), while the BingWorstCase strategy results in routes that are on average 7.4% longer than the shortest solution found. However, the low total times of the route come with high delays at customers. On average, there are 1.82 late arrivals per route with the OpenRoute strategy. This means that almost 20% of customers experience a delay. Next to that, the average lateness per route is 23.45 minutes for the OpenRoute strategy. As it was expected, the BingWorstCase strategy resulted in almost no delays while Simulated Annealing did not accept delays in the first place. The performance of the latter one is therefore quite well, as the solution deviates on average only by 2.9% to the best solution found. Also the BingAverage strategy is promising as on average only 0.69 delays where attained per route with a mean lateness of 2.35 minutes per route which is reasonable low. The averages of the KPIs per strategy are summarized in Table 6.2.

Table 6.2: Average of selected KPIs for cases with 10 Customers

	Strategy				
KPIs	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.7%	2.3%	2.4%	7.4%	2.9%
Lateness in Minutes	23.45	14.12	2.35	0.005	0
Number of times arrived late	1.82	1.28	0.69	0.005	0

20 Customers

Regarding the boxplot 6.3 the results for 20 customers are similar to the ones with 10 customers. It can be seen that the OpenRoute strategy resulted in the shortest routes (as measured by time) but highest delays. The BingWorstCase strategy resulted in the longest routes and almost no delays. Now, we look closer at the averages of these KPIs.

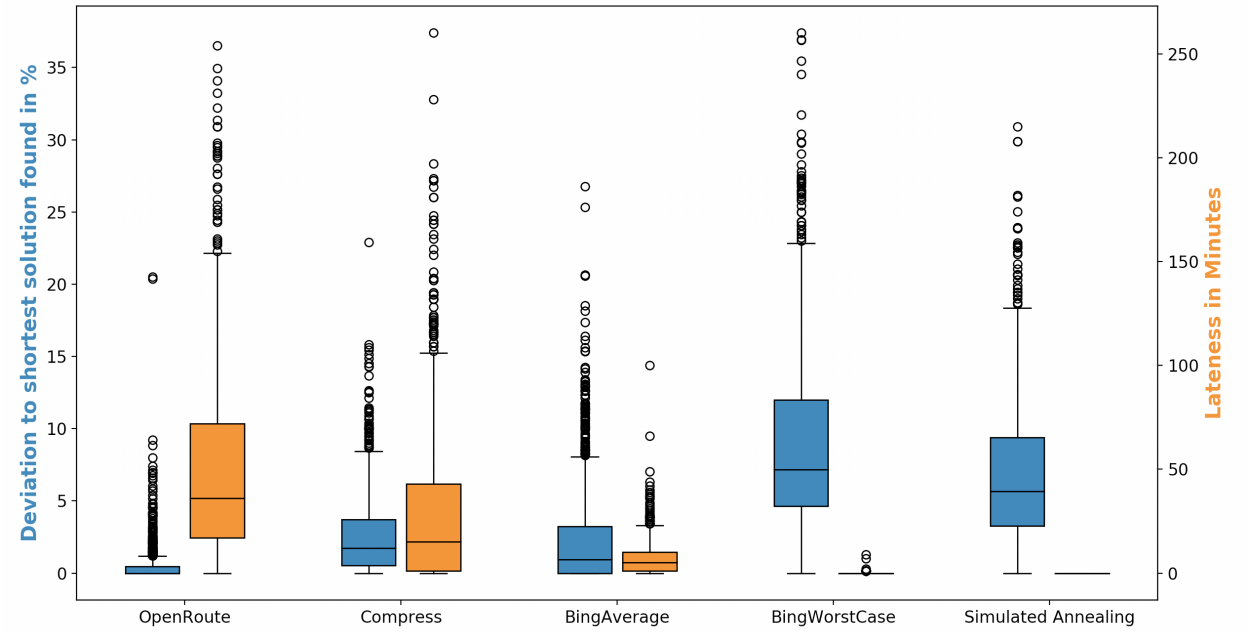


Figure 6.3: Deviation to shortest solution found and lateness in minutes in cases with 20 customers

From 6.3 we can confirm that overall results are similar to the ones with 10 customers. The OpenRoute strategy is on average only 0.5% longer than the shortest solution found. We can see that both BingWorstCase and the Simulated Annealing strategy performed worse compared to the case with 10 customers when looking at the total time (7.4% to 9.1% and 2.9% to 6.83% respectively). The number of delays and the total lateness in minutes per route approximately doubled for the OpenRoute, Compress and BingAverage strategy. In the open route strategy, still around 20% of the customers are served late with an average lateness of 50.37 minutes per route and a respective lateness of $50.37/3.98 = 12.6$ minutes per customer given that the customer is served late. Interesting to see is that the BingAverage strategy beats the Compress strategy on both the total time and the lateness.

Table 6.3: Average of selected KPIs for cases with 20 Customers

	Strategy				
KPIs	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.5%	2.6%	2.4%	9.1%	6.83%
Lateness in Minutes	50.37	29.53	6.98	0.011	0
Number of times arrived late	3.98	2.61	1.61	0.008	0

30 Customers

From the boxplot 6.4 we can see that results are similar to the previous two cases with the exception that the Simulated Annealing strategy now performs worst in terms of total time. As for the other

two cases, there are many outliers for both travel time and lateness for all five strategies.

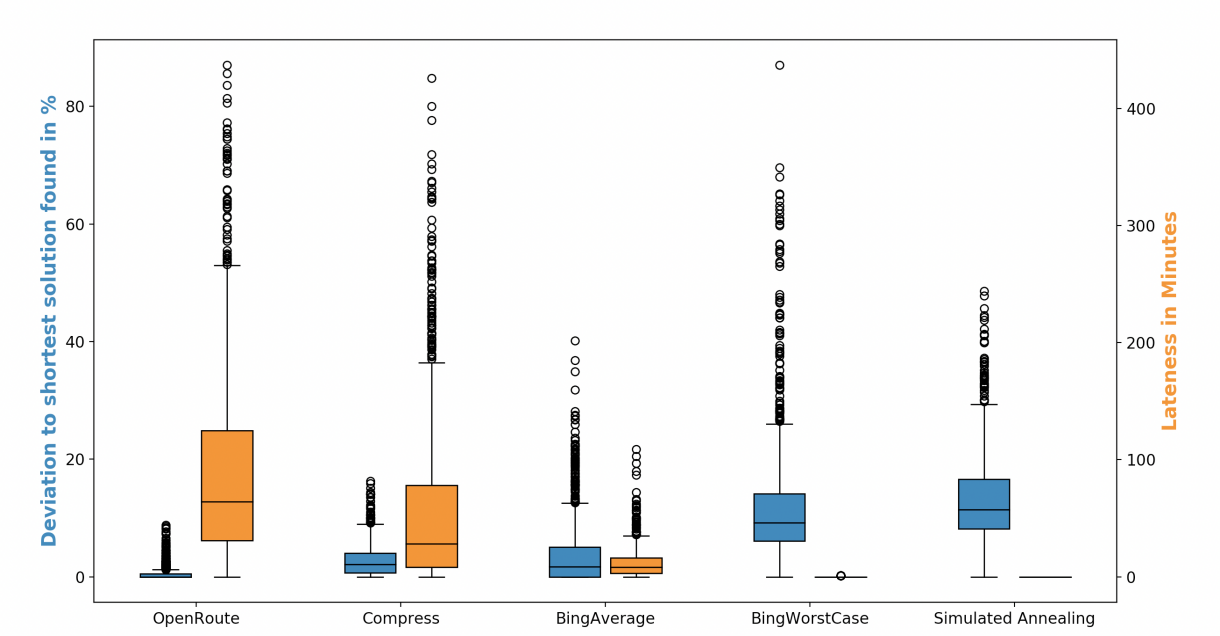


Figure 6.4: Deviation to shortest solution found and lateness in minutes in cases with 30 customers

For the total time of the routes, the OpenRoute strategy still performs best while the Compress strategy now performs second best and the BingAverage strategy third best. Interesting to see is that in terms of total time the Simulated Annealing strategy now performs worse than the BingWorstCase strategy. This is a sign that the Simulated Annealing strategy requires higher run times than the other strategies and therefore only performs well on small sized problems within a reasonable running time. Two major reasons for that behavior are identified: Firstly, the evaluation includes the calculation of third polynomial functions which requires some time and secondly, the program is written in Python as opposed to the Google OR tools algorithms which are written in C++, which is a considerably faster programming language. After the main experiments have been performed, we have increased the algorithm run time (from 30 seconds to 200 seconds) for all strategies, for a few cases with 30 customers. We observed that the performance of the Simulated Annealing strategy increases more than the performance of the other strategies. This confirms the hypothesis that the Simulated Annealing strategy requires higher run times to result in good routing solutions. For the case of 30 customers, it is clearly visible that lower total times come with a higher lateness. The Simulated Annealing strategy naturally results in no delays and the BingWorstCase strategy has almost no delays. For the OpenRoute and Compress strategy, around 21% and 15% of the customers are served late, respectively. For the BingAverage strategy, this value is around 8% which is the same as for the case of 20 and 10 customers. Next to the number of delays also the lateness in minutes, given that a customer is served late, is reasonably lower for the BingAverage strategy compared to the first two strategies, namely 4.33 minutes compared to 12.65 (OpenRoute) and 11.31 (Compress) minutes .

Table 6.4: Average of selected KPIs for cases with 30 Customers

KPIs	Strategy				
	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.5%	2.8%	3.77%	12.2%	13.2%
Lateness in Minutes	91.1	57.84	11.4	0.016	0
Number of times arrived late	6.42	4.48	2.45	0.016	0

6.2.2 Results by Maximum Distance to Depot

We are now looking at the results from a different perspective. Cases with a maximum distance between depot and customers of 100 km, 200 km and 300 km are looked at separately.

100 Kilometer

From Figure 6.5 we can see that the BingAverage strategy beats the OpenRoute and Compress strategy on both, the lateness in minutes as well as on the route length. The BingWorstCase and Simulated Annealing strategy result in considerably longer routes compared to the other strategies, but do come with almost no delays.

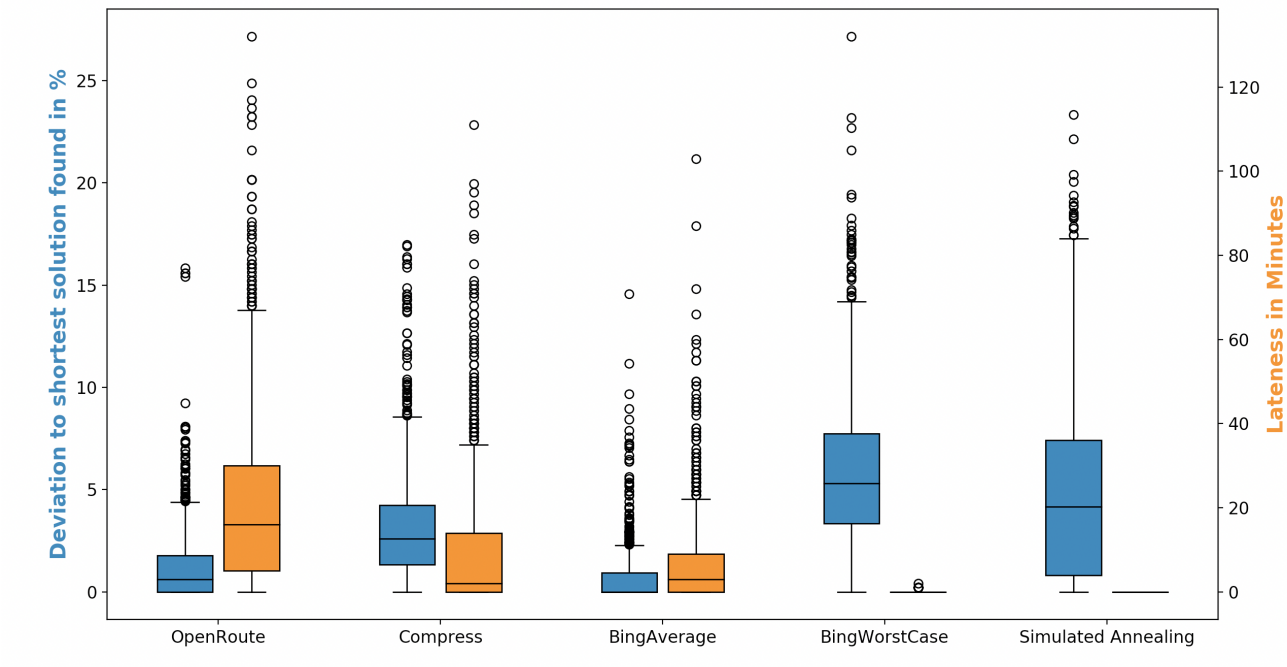


Figure 6.5: Deviation to shortest solution found and lateness in minutes in cases with a maximum distance of 100km to the depot

Table 6.5 confirms these findings. The BingAverage strategy is on average just 0.6% longer than the shortest route found while this value is 1.1% for the OpenRoute and 3.1% for the Compress strategy. The longest routes are attained by the BingWorstCase strategy (6% longer than the shortest route found) and Simulated Annealing (4.7%). The OpenRoute strategy results in the highest delays, measured by both minutes (21.5 minutes on average) and number of times arrived late (2.7). This means that almost 3 customers are delivered late in this strategy and a lateness of around 7 minutes given that a customer is delivered late. The BingAverage strategy beats the OpenRoute and Compress strategy by delays measured in minutes, it results however in a higher total late arrivals (1.6 on average per route) than the Compress strategy (1.4). Simulated Annealing naturally comes with no delays while BingWorstCase has almost no delays either.

Table 6.5: Average of selected KPIs for cases with a maximum distance of 100km to the depot

	Strategy				
KPIs	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	1.1%	3.1%	0.6%	6.0%	4.7%
Lateness in Minutes	21.5	9.9	6.8	0	0
Number of times arrived late	2.7	1.4	1.6	0	0

200 Kilometer

For the cases of a maximum distance of 200 km between depot and customers, Figure 6.6 looks similar to 100 km Figure (6.5) with the difference that this time the OpenRoute strategy results in by far the shortest routes. We therefore now look at the exact numbers from Table 6.6.

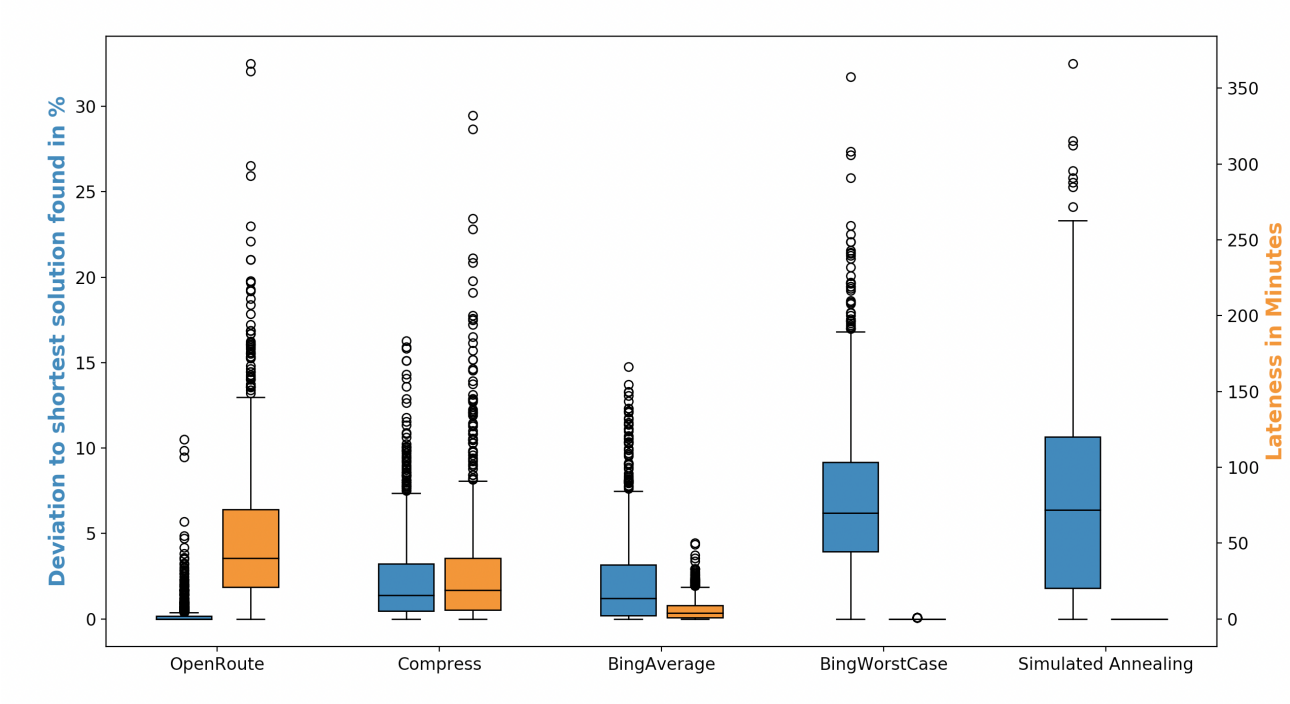


Figure 6.6: Deviation to shortest solution found and lateness in minutes in cases with a maximum distance of 200km to the depot

The OpenRoute strategy results on average in 0.2% longer routes than the shortest route found. The BingAverage and Compress Strategy perform both equally good on this KPI with a value of 2.2%. Both BingWorstCase and Simulated Annealing perform slightly worse than previously regarding the total time (from 6% to 7.2% and from 4.7% to 7.0% respectively). In terms of lateness in minutes, we can see that the OpenRoute strategy still performs worst and more than doubled compared to the 100 km case (from 21.5 to 53.4 minutes on average). This is similar for the compress strategy where the lateness in minutes tripled from 9.9 to 30.5 minutes on average. Also the number of times arrived late increased for these two strategies (from 2.7 to 4.2 and from 1.4 to 2.8 respectively). For the other three strategies, the delay measured by lateness and by the times customers were served late remained similar to the 100 km case.

Table 6.6: Average of selected KPIs for cases with a maximum distance of 200km to the depot

	Strategy				
KPIs	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.2%	2.2%	2.2%	7.2%	7.0%
Lateness in Minutes	53.4	30.5	6.3	0	0
Number of times arrived late	4.2	2.8	1.6	0	0

300 Kilometer

From Figure 6.7 we can see that the BingAverage strategy now performs worse than the Compress strategy in terms of the total route length. Besides that, the Figure appears similar to the ones of the cases with a maximal distance of 100 and 200 km.

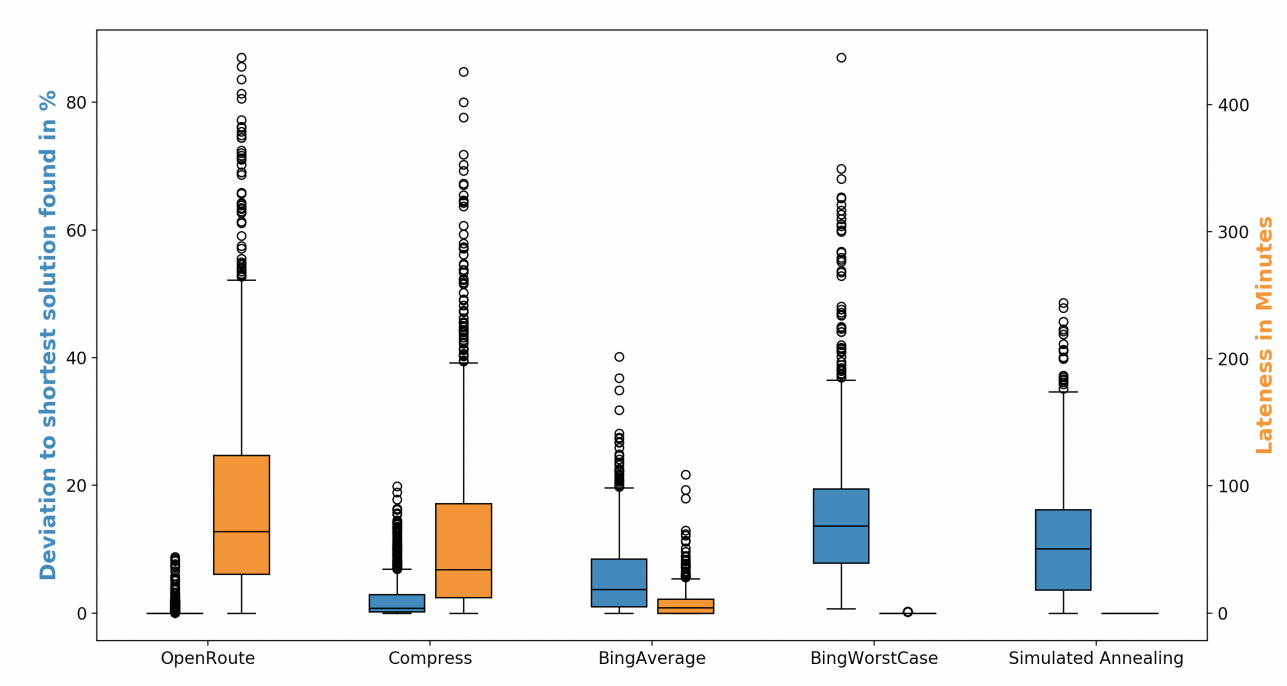


Figure 6.7: Deviation to shortest solution found and lateness in minutes in cases with a maximum distance of 300km to the depot

By looking at Table 6.7 we can immediately see that the trade off between shorter routes and delays increases. While the BingAverage, BingWorstCase and Simulated Annealing result in considerably longer routes (5.6%, 15.5% and 11.1% longer than the shortest route on average), the OpenRoute and Compress strategy result in a considerably high lateness, namely 90.5 and 61 minutes on average respectively, compared to 7.5 minutes for the BingAverage strategy and no delays for the other two strategies.

Table 6.7: Average of selected KPIs for cases with a maximum distance of 300km to the depot

	Strategy				
KPIs	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.2%	2.2%	5.6%	15.5%	11.1%
Lateness in Minutes	90.5	61.0	7.5	0	0
Number of times arrived late	5.3	4.2	1.5	0	0

6.2.3 Results by Time-Window Size

Previously we have looked at the performance of the strategies by splitting up the cases into size (number of customers) and the maximum distance between customers and the depot. We now look at the results attained depending on the time-window sizes.

0.5 Hours

Figure 6.8 shows the results for cases in which every customer had a time-window size of half an hour. We can see that the overall distribution is similar to the distributions that we have already seen in the previous cases. We now further look at the exact numbers which are presented in Table 6.8.

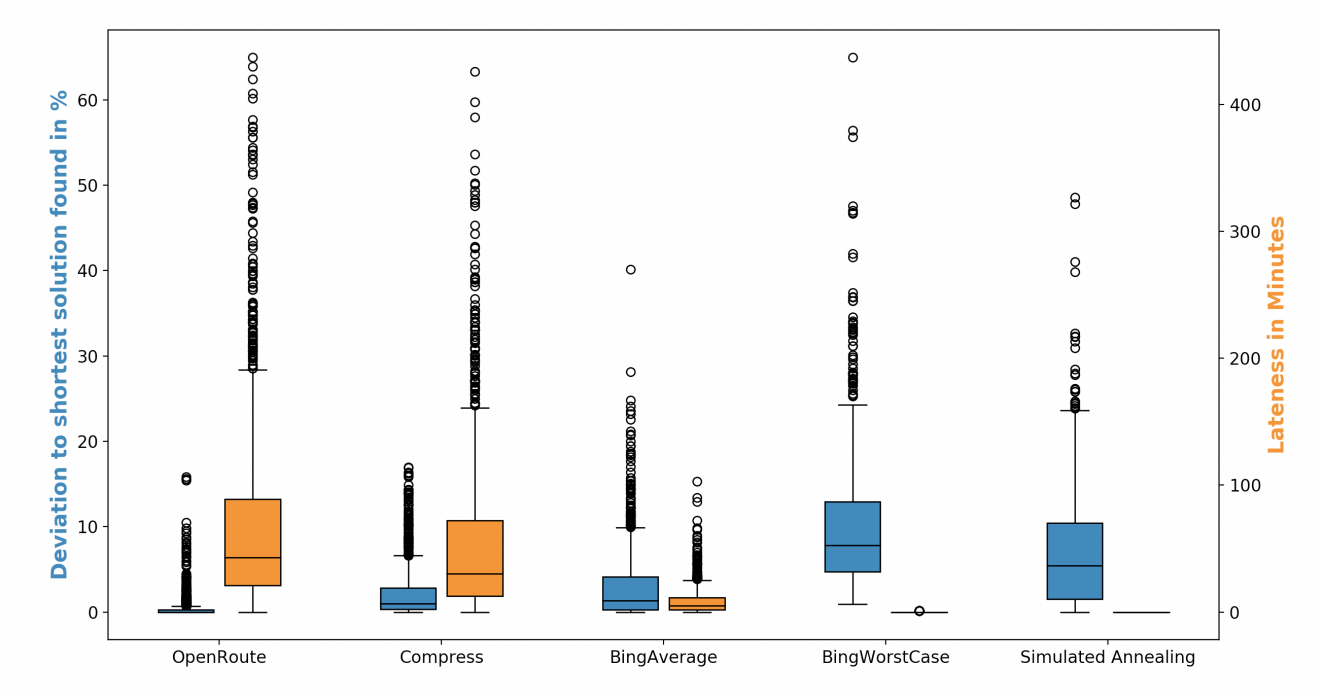


Figure 6.8: Deviation to shortest solution found and lateness in minutes in cases with time windows of 0.5 hours

We can see that the basic trade off between short routes and delays also exists here. The OpenRoute strategy results in short routes (on average only 0.5% longer than the shortest route found), but also in high delays as measured in minutes (70.2) and number of times arrived late (5.6) on average per route. Compared to the Compress strategy, the BingAverage strategy only results in slightly longer routes (3.1% compared to 2.3%) but considerably less delays. The times a customer was served late is less than 50% compared to the Compress strategy and the lateness in minutes decreased from 54.9 to 8.5 minutes. The BingWorstCase strategy resulted in almost no delays but higher travel times. On average, the routes found were almost 10% longer than the shortest solution found while this value is 7% and therefore favorably for the Simulated Annealing strategy.

Table 6.8: Average of selected KPIs for cases with time windows of 0.5 hours

KPIs	**Strategy**				
	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.5%	2.3%	3.1%	9.9%	7.0%
Lateness in Minutes	70.2	54.9	8.5	0	0
Number of times arrived late	5.6	4.6	2.2	0	0

1 Hour

Figure 6.9 presents the results for cases in which every customer has a time-window of one hour. As for the other cases, the OpenRoute strategy results in the shortest routes but highest delays and the BingWorstCase strategy in the longest routes and almost no delays. The BingAverage strategy represents a trade-off between these two extremes.

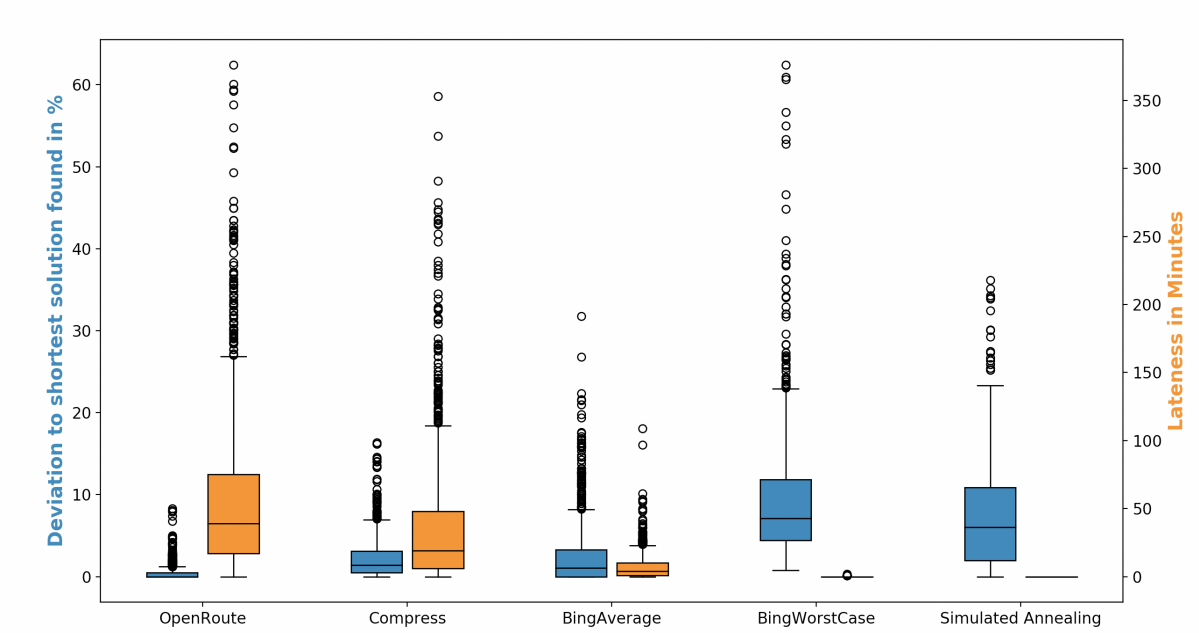


Figure 6.9: Deviation to shortest solution and lateness in min. in cases with time windows of 1 hour

From Table 6.9 we can see that the Simulated Annealing strategy outperforms the BingWorstCase strategy because it has no delays but its routes are shorter (on average 7.2% longer than the shortest solution found compared to 9.3% longer for the BingWorstCase strategy). The OpenRoute strategy results in also one hour of delay and 4.3 delays per route on average. While the BingAverage strategy results in slightly longer routes, it cuts delays by more than 60% in terms of the number of times arrived late and by more than 85% in terms of the lateness in minutes.

Table 6.9: Average of selected KPIs for cases with time windows of 1 hour

KPIs	**Strategy**				
	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.5%	2.2%	2.5%	9.3%	7.2%
Lateness in Minutes	59.2	37.2	7.45	0	0
Number of times arrived late	4.3	3.0	1.7	0	0

3 Hours

For cases where the customers' time-windows are relatively long (3 hours), we can see that naturally all strategies result in less delays than for shorter time-windows. Figure 6.10 shows these results.

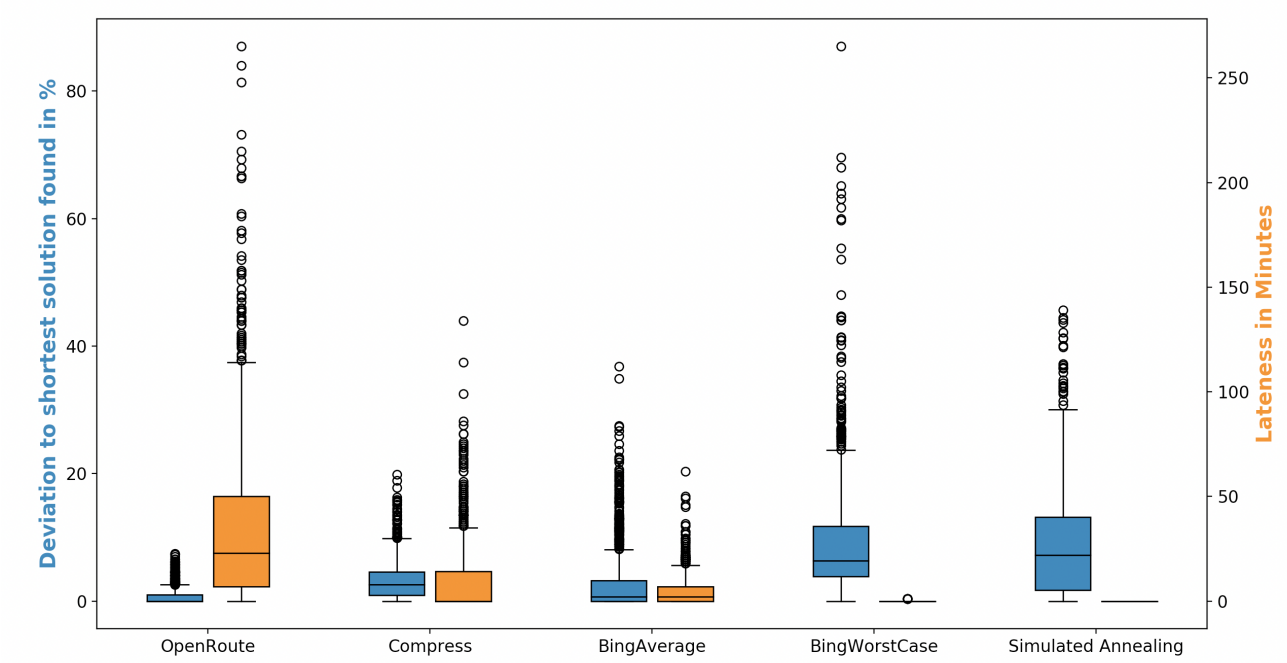


Figure 6.10: Deviation to shortest solution found and lateness in minutes in cases with time windows of 3 hours

From Table 6.10 we can see that the OpenRoute strategy resulted again in the shortest routes but highest delays. The Compress and BingAverage strategy perform similarly in both route length and the number of late arrivals. In terms of the lateness measured in minutes, the BingAverage strategy however results in 50% less delays. The BingWorstCase strategy results in the longest routes and almost no delays while the Simulated Annealing strategy results in shorter routes and naturally no delays.

Table 6.10: Average of selected KPIs for cases with time windows of 3 hours

	Strategy				
KPIs	OpenRoute	Compress	BingAverage	BingWorstCase	Simulated Annealing
Total Time: Deviation to shortest solution found	0.7%	3.2%	2.9%	9.5%	8.7%
Lateness in Minutes	35.6	9.5	4.7	0	0
Number of times arrived late	2.3	0.8	0.9	0	0

6.3 Result Validation

The above mentioned results are based on the key assumption that the time-dependent data of Bing is a realistic representation of the true travel times. In order to validate these results, we evaluate part of the routes with a third data-source, namely data from the Google Distance Matrix API (*Google Distance Matrix API*, 2020). As the Google API has lower request limits, not all instances can be evaluated by Google data. We therefore choose three instances with ten customers and three instances with 20 customers and source time-dependent traveltimes for these instances on Friday the 31st July 2020 (as it has been done with Bing data). As before, we have three different time-window settings and ten replications. The instances are then solved by the five strategies and evaluated by both, Bing and Google data.
Figure 6.11 shows how the strategies deviated from the best solutions found. We can see that the boxplots attained with Google and Bing data look very similar. In fact, the OpenRoute strategy results in the shortest routes in both cases. The Compress and BingAverage strategy perform equally good. The BingWorstCase strategy results in the longest routes while Simulated Annealing results in the second longest routes on average.

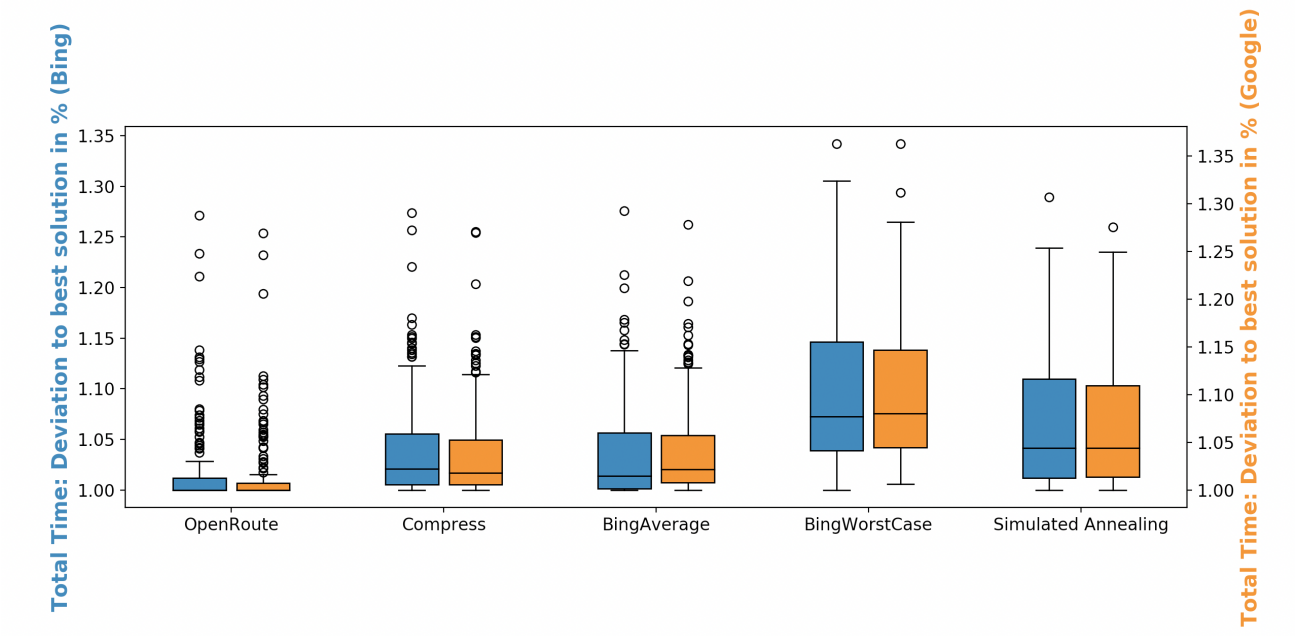


Figure 6.11: Deviation to shortest solution found by Bing and Google

Also the lateness in minutes is similar for both data sources (see Figure 6.12). The OpenRoute strategy and Compress strategy result in the highest delays. Interesting to see is that the BingAverage, BingWorstCase and Simulated Annealing strategy have higher delays with the Google data.

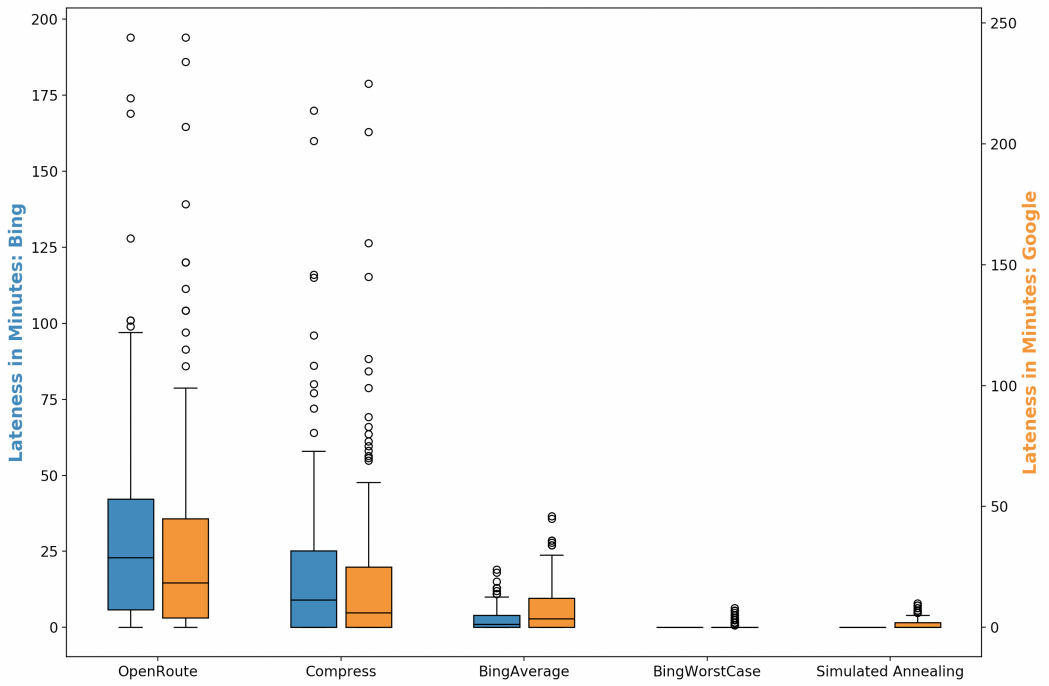


Figure 6.12: Lateness in minutes by Bing and Google

As for the previous two KPIs, also the boxplots for the number of late arrivals look very similar for both data sources. We can however see that for the Simulated Annealing strategy, the number of late

arrivals has a median of one with Google data, compared to almost no delays for the BingWorstCase strategy. This is interesting as it seems that the Simulated Annealing strategy overfits on the Bing data and gives less robust routes than the BingWorstCase strategy on Google data. It might therefore be advisable not to optimize routes solely on (deterministic) time-dependent travel time data without improving the robustness of the routes with rather stochastic time-dependent data or multiple data sources.

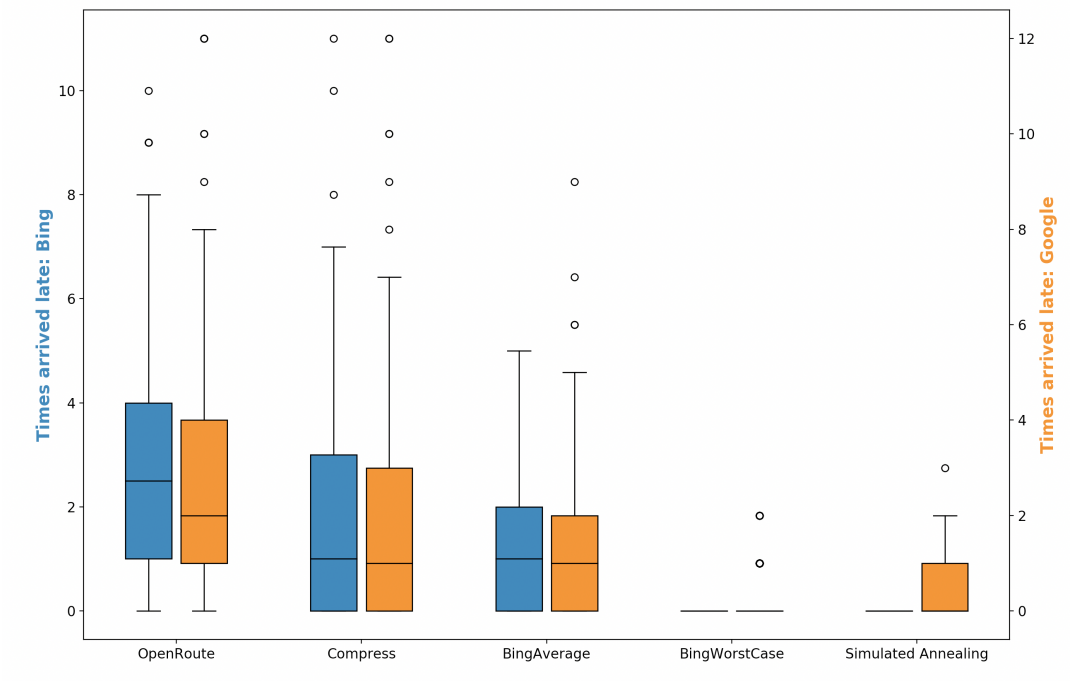


Figure 6.13: Number of late arrivals by Bing and Google

6.4 Conclusion

The first research question this chapter answered was 'How can the collected data be used to model realistic VRP instances?'. Therefore, we have first constructed problem instances with different settings, namely different number of customers (10, 20 and 30) and a different maximum distance between customers and the depot (100 km, 200 km and 300 km) from the data presented in Chapter 4. For each of these settings, four different instances have been created. The next research questions were 'To what degree does predictable traffic intensity influence current routing plannings in terms of the chosen KPIs?' and 'How do the tested strategies perform in terms of the chosen KPIs?'. In order to answer these two research questions, the five strategies presented in Chapter 4 have then been applied to solve the instances. After that, routes and departure times the strategies have generated have been collected and lastly these routes have been evaluated under time dependent travel times. The results have been split up by problem size (the number of customers), the maximum distance between customers to the depot and by time-window size. The results for all these separations are similar. In general there is a trade-off between the route length and the number of delays: Shorter routes (as measured in time) come with higher delays. The currently used OpenRoute strategy resulted in the shortest routes but by far highest delays: Around 20% of the customers are served late. These delays can be cut by more than 60% by taking daily averages of travel times (BingAverage strategy) with an average increase of the travel times by only around 2%. The BingAverage strategy therefore seems to be a very good alternative to the current strategy applied (using OpenRoute data). The Simu-

lated Annealing strategy (A-strategy) performed very well for smaller cases of 10 customers but worse for bigger cases. For 30 customers, the BingWorstCase strategy even outperformed the Simulated Annealing strategy on the total route length. Two reasons for the bad performance on larger cases are determined: Firstly, the evaluation includes the calculation of third polynomial functions which requires some time and secondly, the program is written in Python as opposed to the Google OR tools algorithms which are written in C++, which is a considerably faster programming language. The trade off between shorter routes (as measured in time) and delays increases when the vehicles need to travel a longer distance (i.e. if the maximum distances between customers and the depot increase). In other words, for the cases vehicles need to travel a long distance the OpenRoute and Compress strategy result in shorter routes but higher delays than the other three strategies. In cases where the maximum distance between customers and depot is 100 kilometers, that is in cases where customers are more densely clustered, the BingAverage strategy does not only result in the shortest routes but also beats the OpenRoute and Compress strategy on delays. As it could have been expected, delays increase when the time-window size decreases. This increase is however higher for the OpenRoute and Compress strategy. The last research question in this chapter was 'How can the results be validated?'. In order to answer this question, we used the Google distance matrix API and compared the results attained from Google and Bing data for three instances with 10 customers and three instances with 20 customers (the free Google API limits did not allow for more instances). We found that the results for both, route length as well as delays is very similar for both data sources, which strengthened the validity of the found results. However, the Simulated Annealing strategy resulted in substantially more delays than the BingWorstCase strategy when evaluating the routes with Google data. It can be concluded that the Simulated Annealing strategy overfits on the Bing data and gives less robust routes than the BingWorstCase strategy on Google data. For the WDL, it is therefore advisable not to optimize routes solely on (deterministic) time-dependent travel time data without improving the robustness of the routes with rather stochastic time-dependent data or multiple data sources.

In conclusion it can be stated that better routes can be obtained by using time-dependent travel times. Even a relatively simple strategy such as just computing a travel time matrix which contains the daily average of all directed location pairs (BingAverage strategy) results in considerably better routes. Delays can be cut by more than 60% with an average increase of the travel time by only around 2%. The BingWorstCase strategy (taking the longest travel time (worst-case) for the specific day for each directed location pair) results in the most robust solutions.

CHAPTER
SEVEN

CONCLUSION AND RECOMMENDATIONS

This chapter is split up into four different parts. Firstly, a conclusion for the overall research is provided. Afterwards, limitations of the study are explained and further research is suggested. Lastly, we provide an answer to the question 'Which strategies are most favorable to put into practice by the WDL?' by providing a recommendation to the WDL.

7.1 Conclusion

In Chapter 1, we have introduced our main research question as: 'How and to what extent can the WDL incorporate historical traffic data to provide better routing plannings in terms of the chosen KPIs?'. We have shown that using time-dependent travel times, current routes of the WDL can be substantially improved. More precisely, even a relatively simple strategy such as just computing a travel time matrix which contains the daily average of all directed location pairs (BingAverage strategy) results on average in a decrease of delays of more than 60% with just a slight increase of the route duration (around 2%) compared to the current strategy used by the WDL (using time-independent travel times of the OpenRoute API). This conclusion was derived as follows: In Chapter 2, background knowledge about the VRP was provided. In Chapter 3, a literature review on the Time Dependent Vehicle Routing Problem (TDVRP) was performed. In literature, various strategies with different levels of congestion avoidance have been developed to improve routes with time-dependent travel times. Chapter 4 includes the research framework of this study. The research framework consists of three levels: The input, model and output level. The input to the model has been explained by the presentation of geographical location data as well as the retrieval of travel times and the interpolation of travel times. The model itself was explained by introducing the strategies that were tested and the algorithm that is used to evaluate routing plannings. Thirdly, the output of the model, namely the KPIs is presented. In Chapter 5, travel-time test data was presented and characteristics of this data was analyzed. This information was then used to construct realistic VRP instances, which is done in Chapter 6. In total, 36 instances (set of depot and customers locations) were determined. Next to the construction of these instances, the five strategies described in Chapter 4 were used to solve these instances with various time-window schedules. It was then evaluated how the routes perform under time dependent travel times. After presenting the results, a validation of those results with a second data-source for time dependent travel times was conducted for six of these 36 instances, namely with data from the 'Google distance matrix API'.

7.2 Limitations

Our results are based on the data that we sourced. Despite the fact that we increased the validity of the results that we observed with our first data source (Bing distance matrix API data) by a second data source (Google's distance matrix API), the research still depends on the assumption that these travel times are realistic representations of the reality. Using additional data sources or preferably testing the different routes in reality would help increasing the validity of this research. Next to that, we have only conducted experiments with one type of the VRP with specific constraints such as solely a time-minimization objective and hard time-windows. Even so, we expect the results to be comparable for other types of the VRP, as the general problem of traffic congestion is also valid for other types of the VRP. However, different constraints and objectives might influence the results and might lead to different trade-offs.

7.3 Future Research

The future research is tightly coupled to the limitations of the study. The suggested strategies could be tested on different data and possibly even tested in real life. Next to that, VRP models with different objective values and constraints could be tested. More concrete, we identify two interesting additions to our VRP model that should be considered in future studies. Firstly, it would be interesting to include soft time-windows, that is, late arrivals are allowed and come with a certain penalty. Secondly, the minimization of the number of vehicles should be incorporated as this is also a common objective in many variations of the VRP. Next to that, more strategies that incorporate time-dependent traffic data could be tested. For example, it would be interesting to combine the 'BingAverage' strategy with a safety slack for highly variable routes to reduce delays. Similarly to the Simulated Annealing algorithm implemented in this research, new algorithms could be developed which include time-dependent travel times directly. These algorithms should however also take the robustness of routes into account. This could be done by some kind of approach that includes a sensitivity analysis or a simulation approach with stochastic time-dependent travel times.

7.4 Recommendations

The last sub- research question asked in this study was the following: 'Which strategies are most favorable to put into practice by the WDL?'. Generally it can be stated that routes always come with a time-delay trade off. Given a number of customers that have to be served from a depot, routes with no delays take more time than routes which come with delays. However, it has been shown that the application of a travel time matrix which contains the daily average of all directed location pairs (BingAverage strategy) resulted in considerably better routes compared to the current strategy used. More precisely, delays can be cut by more than 60% with an average increase of the travel time by only around 2%. As the data of the Bing distance matrix API provides a generous free of charge usage option of 500,000 requests per year (where one request is one location-pair distance for one departure time), we advise the WDL to implement this strategy (BingAverage) in order to improve the routes they provide to their clients without having to carry out a large investment or other greater efforts. In cases where delays must be eliminated at all costs, the BingWorstCase strategy (taking the longest travel time (worst-case) for the specific day for each directed location pair) results in the most robust solutions and should therefore be applied.

REFERENCES

Applegate, D. L., Bixby, R. E., Chvatál, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study.* Princeton University Press.

Balseiro, S., Loiseau, I., & Ramonet, J. (2011). An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers & Operations Research*, *38*(6), 954 - 966. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0305054810002327` doi: https://doi.org/10.1016/j.cor.2010.10.011

Bing distance matrix api. (2020). Retrieved from `https://docs.microsoft.com/en-us/bingmaps/rest-services/routes/distance-matrix-data`

Braekers, K., Ramaekers, K., & Nieuwenhuyse, I. V. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300-313. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0360835215004775` doi: https://doi.org/10.1016/j.cie.2015.12.007

Bundesverband Güterkraftverkehr Logistik und Entsorgung BGL e.V. (2020). *Europäisches Übereinkommen über die arbeit des im internationalen straßenverkehr beschäftigten fahrpersonals (aetr) vom 01.07.1970 (bgbl. ii s. 1475 vom 20.12.1974) zuletzt geändert durch das gesetz vom 02.11.2011 (bgbl. ii s. 1095).* (https://www.bgl-ev.de/images/downloads/service/gesetze/aetr.pdf)

Christofides, N., Mingozzi, A., & Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, *20*, 255-282.

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

Dabia, S., Ropke, S., van Woensel, T., & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, *47*(3), 380-396. Retrieved from `https://pubsonline.informs.org/doi/abs/10.1287/trsc.1120.0445` doi: 10.1287/trsc.1120.0445

Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., & Gambardella, L. M. (2008). Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, *185*(3), 1174 - 1191. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0377221706006345` doi: https://doi.org/10.1016/j.ejor.2006.06.047

Encyclopædia britannica. (2011). Encyclopædia Britannica, inc. Retrieved from `https://www.britannica.com/science/annealing-heat-treatment`

Glover, F., & McMillan, C. (1986). The general employee scheduling problem. an integration of ms and ai. *Computers & Operations Research*, *13*(5), 563 - 573. Retrieved from `http://www.sciencedirect.com/science/article/pii/030505488690050X` (Applications of Integer Programming) doi: https://doi.org/10.1016/0305-0548(86)90050-X

Google distance matrix api. (2020). Retrieved from `https://developers.google.com/maps/documentation/distance-matrix/overview`

Gromicho, J., van Hoorn, J., Kok, A., & Schutten, J. (2012). Restricted dynamic programming: A flexible framework for solving realistic vrps. *Computers & Operations Research*, *39*(5), 902 - 909. Retrieved from `http://www.sciencedirect.com/science/article/pii/S030505481100195X` doi: https://doi.org/10.1016/j.cor.2011.07.002

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, *144*(2), 379 - 396. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0377221702001479` doi: https://doi.org/10.1016/S0377-2217(02)00147-9

Jung, S., & Haghani, A. (2001). Genetic algorithm for the time-dependent vehicle routing problem. *Transportation Research Record*, *1771*(1), 164-171. Retrieved from `https://doi.org/10.3141/1771-21` doi: 10.3141/1771-21

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. Retrieved from `https://science.sciencemag.org/content/220/4598/671` doi: 10.1126/science.220.4598.671

Kok, A., Hans, E., & Schutten, J. (2012). Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & operations research*, *39*(5), 910–918. doi: 10.1016/j.cor.2011.05.027

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43*(4), 408-416. doi: 10.1287/trsc.1090.0301

Laporte, G., Toth, P., & Vigo, D. (2013). Vehicle routing: historical perspective and recent contributions. *EURO J Transp Logist*, *2*, 1-4. doi: 10.1007/s13676-013-0020

Law, A. M. (2005). How to build valid and credible simulation models. In *Proceedings of the winter simulation conference, 2005.* (p. 9 pp.-).

Lecluyse, C., Van Woensel, T., & Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times [Article]. *4OR - A Quarterly Journal of Operations Research*, *7*(4), 363-377. doi: {10.1007/s10288-009-0097-9}

Lombard, A., Tamayo Giraldo, S., & Fontane, F. (2018). Modelling the time-dependent vrp through open data. In I. M. of Engineers & C. Scientists (Eds.), .

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

Malek, M., Guruswamy, M., & Pandya, M. (1990). Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Ann. Oper. Res.*, *21*(1–4), 59–84. Retrieved from `https://doi.org/10.1007/BF02022093` doi: 10.1007/BF02022093

Mancini, S. (2014). Time dependent travel speed vehicle routing and scheduling on a real road network: The case of torino. *Transportation Research Procedia*, *3*, 433 - 441. Retrieved from http://www.sciencedirect.com/science/article/pii/S2352146514001872 (17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain) doi: https://doi.org/10.1016/j.trpro.2014.10.024

Nahum, O. E., & Hadas, Y. (2009). Developing a model for the stochastic time-dependent vehicle-routing problem. In *2009 international conference on computers industrial engineering* (p. 118-123).

Openrouteservice matrix. (2020). Retrieved from https://openrouteservice.org/dev/#/api-docs/matrix/

Osvald, A., & Stirn, L. Z. (2008). A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food. *Journal of Food Engineering*, *85*(2), 285 - 295. Retrieved from http://www.sciencedirect.com/science/article/pii/S0260877407004141 doi: https://doi.org/10.1016/j.jfoodeng.2007.07.008

Perron, L., & Furnon, V. (2019). *Or-tools.* Retrieved from https://developers.google.com/optimization/

Savelsbergh, M. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, *4*(2), 146-154. Retrieved from https://doi.org/10.1287/ijoc.4.2.146 doi: 10.1287/ijoc.4.2.146

Savelsbergh, M., & Van Woensel, T. (2016). City Logistics: Challenges and Opportunities. *Transportation Science*, *50*(2, SI), 579-590.

Skabardonis, A., Varaiya, P., & Petty, K. F. (2003). Measuring recurrent and nonrecurrent traffic congestion. *Transportation Research Record*, *1856*(1), 118-124. doi: 10.3141/1856-12

Tas, D., Dellaert, N., van Woensel, T., & de Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C-Emerging Technologies*, *48*, 66-83.

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

Wang, X., Choi, T., Li, Z., & Shao, S. (2019). An effective local search algorithm for the multi-depot cumulative capacitated vehicle routing problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1-11.

Woodard, D., Nogin, G., Koch, P., Racz, D., Goldszmidt, M., & Horvitz, E. (2017). Predicting travel time reliability using mobile phone gps data. *Transportation Research Part C: Emerging Technologies*, *75*, 30 - 44. Retrieved from http://www.sciencedirect.com/science/article/pii/S0968090X16302042 doi: https://doi.org/10.1016/j.trc.2016.10.011

APPENDIX
A

APPENDIX

A.1 Google OR-Tools Code for CVRPTW

```
from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp

class GoogleOR:

    def __init__(self, data, timelimit):
        # Just the data is needed to initialize
        self.data = data
        self.timelimit = timelimit

    def solve(self):
        data = self.data
        """Solve the VRP with time windows."""
        manager = pywrapcp.RoutingIndexManager(len(data['time_matrix']),
                                               data['num_vehicles'], data['depot'])
        routing = pywrapcp.RoutingModel(manager)

        def time_callback(from_index, to_index):
            """Returns the travel time between the two nodes."""
            # Convert from routing variable Index to time matrix NodeIndex.
            from_node = manager.IndexToNode(from_index)
            to_node = manager.IndexToNode(to_index)
            return data['time_matrix'][from_node][to_node]

        transit_callback_index = routing.RegisterTransitCallback(time_callback)
        routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)
        time = 'Time'
        routing.AddDimension(
            transit_callback_index,
            600,  # allow waiting time
            1800,  # maximum time per vehicle
            False, # Don't force start cumul to zero.
            time)
        time_dimension = routing.GetDimensionOrDie(time)
        time_dimension.SetSpanCostCoefficientForAllVehicles(1000)
```

```
# Add Capacity constraint.
def demand_callback(from_index):
    """Returns the demand of the node."""
    # Convert from routing variable Index to demands NodeIndex.
    from_node = manager.IndexToNode(from_index)
    return data['demands'][from_node]

demand_callback_index = routing.RegisterUnaryTransitCallback(
    demand_callback)
routing.AddDimensionWithVehicleCapacity(
    demand_callback_index,
    0,  # null capacity slack
    data['vehicle_capacities'], # vehicle maximum capacities
    True,  # start cumul to zero
    'Capacity')

# Add time window constraints for each location except depot.
for location_idx, time_window in enumerate(data['time_windows']):
    if location_idx == 0:
        continue
    index = manager.NodeToIndex(location_idx)
    time_dimension.CumulVar(index).SetRange(time_window[0], time_window[1])

# Add time window constraints for each vehicle start node.
for vehicle_id in range(data['num_vehicles']):
    index = routing.Start(vehicle_id)
    time_dimension.CumulVar(index).SetRange(data['time_windows'][0][0],
                                            data['time_windows'][0][1])
    time_dimension.SetSpanUpperBoundForVehicle(600, vehicle_id) # This line limits
        vehicle driving time

for i in range(data['num_vehicles']):
    routing.AddVariableMinimizedByFinalizer(
        time_dimension.CumulVar(routing.End(i)))



search_parameters = pywrapcp.DefaultRoutingSearchParameters()
search_parameters.first_solution_strategy = (
    routing_enums_pb2.FirstSolutionStrategy.AUTOMATIC)
search_parameters.local_search_metaheuristic = (
    routing_enums_pb2.LocalSearchMetaheuristic.TABU_SEARCH)
search_parameters.time_limit.seconds = self.timelimit
solution = routing.SolveWithParameters(search_parameters)
print("Solver status: ", routing.status())

# Passing solution to other functions in this class
self.solution = solution
self.data = data
self.routing = routing
self.manager = manager
```

A.2 Traffic Density Variation: Tuesday-Thursday

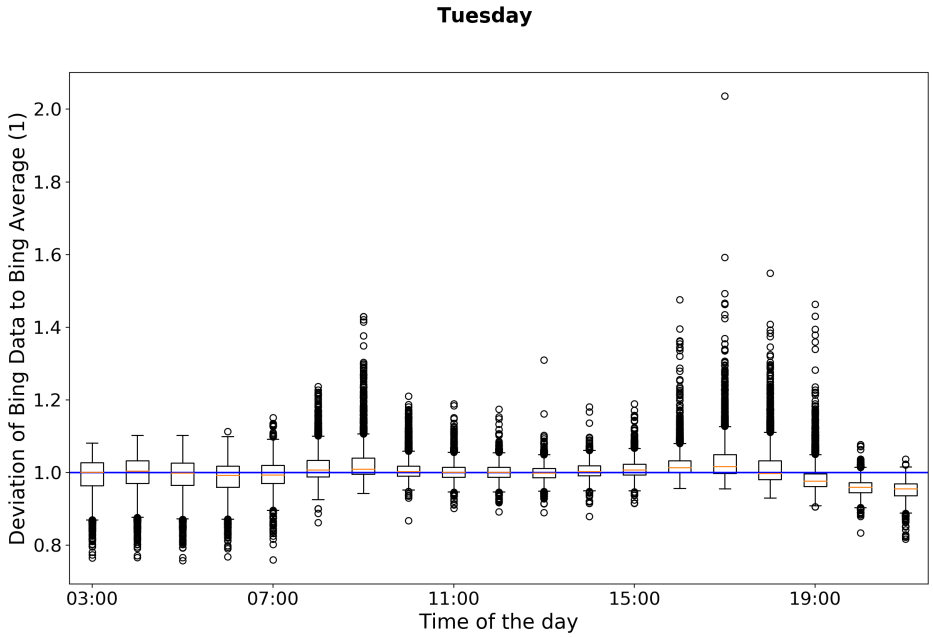


Figure A.1: Traffic Density Variation - Tuesday

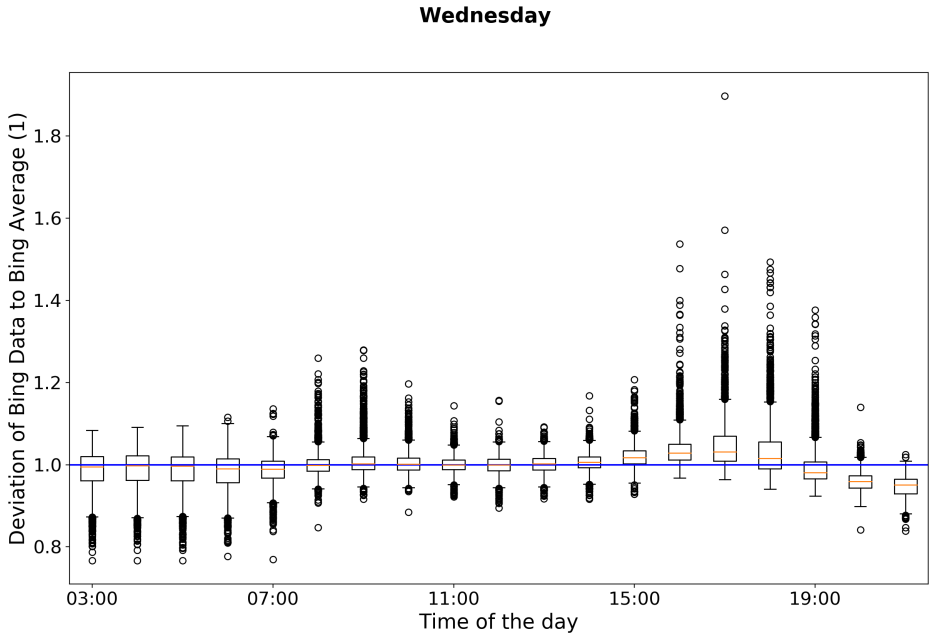


Figure A.2: Traffic Density Variation - Wednesday

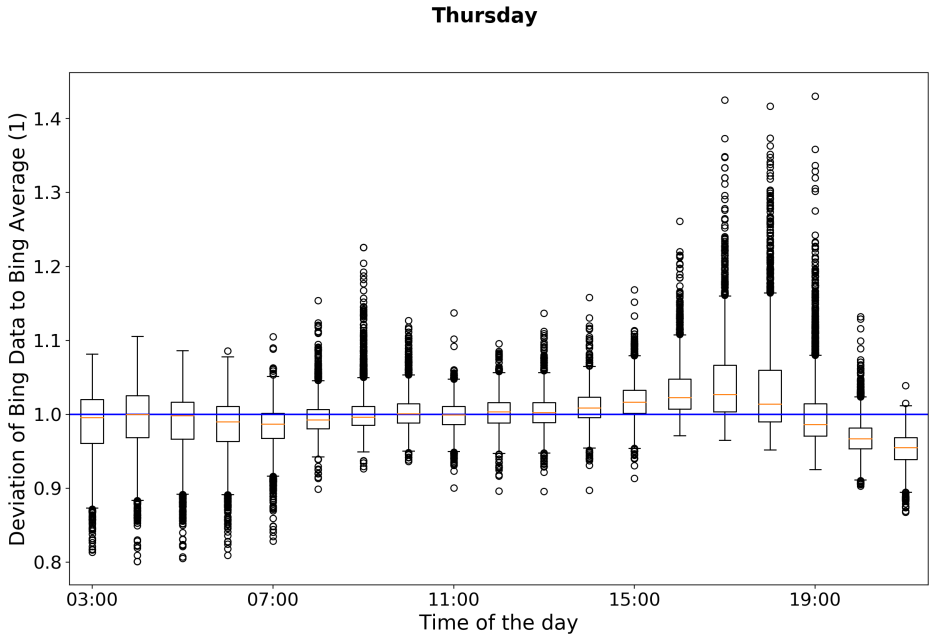


Figure A.3: Traffic Density Variation - Thursday