

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Medieninformatik

Bachelorarbeit

von

Arina **Antskaitis**

**Routenoptimierung für deutschlandweite Dienstleistungen im
betrieblichen Gesundheitsmanagement**

Route optimization for Germany-wide services in occupational health
management

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Medieninformatik

Bachelorarbeit

von

Arina **Antskaitis**

Routenoptimierung für deutschlandweite Dienstleistungen im betrieblichen Gesundheitsmanagement

Route optimization for Germany-wide services in occupational health management

Bearbeitungszeitraum: von 17.10.2022
bis 16.03.2023

1. Prüfer: Prof. Dr.-Ing. Christoph P. Neumann

2. Prüfer: Prof. Dr.-Ing. Gerald Pirkl

Selbstständigkeitserklärung

Name und Vorname

der Studentin/des Studenten:

Antskaitis, Arina

Studiengang:

Medieninformatik

Ich bestätige, dass ich die Bachelorarbeit mit dem Titel:

Routenoptimierung für deutschlandweite Dienstleistungen im betrieblichen Gesundheitsmanagement

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 16.03.2023

Unterschrift:

Bachelorarbeit Zusammenfassung

Studentin/Student (Name, Vorname):	Antskaitis, Arina
Studiengang:	Medieninformatik
Aufgabensteller, Professor:	Prof. Dr.-Ing. Christoph P. Neumann
Durchgeführt in (Firma/Behörde/Hochschule):	wellabe GmbH
Betreuer in Firma/Behörde:	Dr. Sebastian Dünnebeil
Ausgabedatum:	17.10.2022
Abgabedatum:	16.03.2023

Titel:

Routenoptimierung für deutschlandweite Dienstleistungen im betrieblichen Gesundheitsmanagement

Zusammenfassung:

Im Rahmen der vorliegenden Arbeit wird ein Lösungsansatz für die Optimierung der Routenplanung innerhalb einer Firma im gesundheitlichen Bereich entwickelt. Das zugrundeliegende entfernungsbegrenzte Routenplanungsproblem wird auf das notwendige Minimum reduziert und schrittweise gelöst. Auf Basis der Unternehmensdaten werden die möglichen Implementierungsoptionen untereinander verglichen und mit Testszenarien getestet. Die Ein- und Ausgabe erfolgt mittels einer benutzerfreundlichen Weboberfläche. Die Implementierung erstellt einen optimalen Routenplan innerhalb einer angemessenen Zeitspanne vor. Die Kostenfunktion und die Gesamtlänge der gefahrenen Strecken wird minimiert. Auf dessen Basis ist die Implementierung für weitere Entwicklung und Anpassung an geschäftliche Zwänge und Anforderungen möglich.

Schlüsselwörter: TSP, VRP, Routenoptimierung, Operational Research

Abstract:

In the context of the present work, a solution approach for the optimization of route planning within a company in the health management sector is developed. The underlying distance-constrained route planning problem is reduced to the necessary minimum and solved step by step. Based on the company data, the possible implementation options are compared with each other, and test scenarios are built. Interaction with the program, as well input and output are made possible by a user-friendly web interface. The implementation pre-determines an optimal route plan within a reasonable period of time. The cost function and the total length of the driven routes are minimized. Based on this, the implementation is ready for further development and adaptation to business constraints and requirements.

Keywords: TSP, VRP, Route Optimization, Operational Research

Table of Contents

List of Abbreviations	9
List of Figures	10
List of Tables	11
1. Introduction.....	12
1.1. Goal of the Thesis	13
1.2. Research Structure.....	14
2. Methodology.....	17
2.1. On the Tour Planning Problem.....	19
2.2. Route Planning for Multiple Start Depots and Capacitated Resources.....	19
2.3. Complexity Analysis	21
2.3.1. First Case	23
2.3.2. Second Case	24
2.3.3. Third Case	25
2.4. VRP Model.....	26
2.5. Model Relaxation.....	31
2.6. Heuristics for the Optimisation Problem	32
2.7. Clustering Approach	35
2.7.1. OPTICS	41
2.7.2. K-Means	42
2.7.3. Gaussian Mixture Model	45
2.7.4. Agglomerative Clustering.....	47
2.7.5. Clustering Choice	49
2.8. Route Construction.....	49
2.9. Local optimization.....	50
3. Technologies for the Implementation	53
4. Solution Approach.....	56
4.1. Route with no Constraints	56
4.2. Date Assignment	59
4.3. Route with more Nodes	60
4.3.1. Planning a Weekly Route	60
4.3.2. Local Paths' Optimization	62
5. Outcomes.....	71

References	73
Appendices	78
Appendix A: TSP Solver	78
Appendix B: KNN Algorithm	79
Appendix C: Random VRP Path Generation	80
Appendix D: UI of the Web Application	81

List of Abbreviations

AIC Akaike Information Criterion

BIC Bayesian Information Criterion

CVRP Capacitated Vehicle Routing Problem

DB Database

DBI Davis-Bouldin Index

DCVRP Distance-Constrained Vehicle Routing Problem

GMM Gaussian Mixture Model

ILP Integer Linear Programming

IP Integer Programming

MDVRP Multiple Depots Vehicle Routing Problem

MILP Mixed Integer Linear Programming

MIP Mixed Integer Programming

NNA Nearest Neighbor Algorithm

OR Operation Research

RVRP Rich Vehicle Routing Problem

TSP Travelling Salesman Problem

UI User Interface

VRP Vehicle Routing Problem

VRPPD Vehicle Routing Problem with Pickup and Delivery

VRP-SL Vehicle Routing Problem with Service Level Constraints

VRPTW Vehicle Routing Problem with Time Windows

List of Figures

Figure 1 Difference between (left) TSP and (right) VRP	12
Figure 2 Stages of thesis	14
Figure 3 VRP basic scheme	19
Figure 4 Trip for the first case.....	23
Figure 5 Trip for the second case	24
Figure 6 Trip for the third case	25
Figure 7 Case with factors affecting the routes' feasibility	29
Figure 8 Clustering phase.....	36
Figure 9 Routing phase.....	36
Figure 10 Customers' distribution	37
Figure 11 Alternative depot distribution.....	37
Figure 12 Cluster preparation process	38
Figure 13 Depots' distribution together with K-Means centroids.....	39
Figure 14 Clustering and optimization phases	39
Figure 15 OPTICS clustering result	42
Figure 16 Elbow method	43
Figure 17 K-means++ clustering result.....	44
Figure 18 GMM clustering result.....	46
Figure 19 GMM clustering with pre-set centroids	47
Figure 20 Agglomerative clustering results.....	48
Figure 21 Agglomerative clustering with pre-set centroids.....	48
Figure 22 Initial unoptimized route.....	51
Figure 23 Route after 2-opt swap	51
Figure 24 Optimization framework.....	53
Figure 25 Nodes location.....	57
Figure 26 Distance matrix.....	57
Figure 27 Result of TSP approach.....	58
Figure 28 Distance matrix from API response.....	58
Figure 29 Duration matrix from API response	58
Figure 30 Generated use-case plan	59
Figure 31 Plan with opt-in weekends	59
Figure 32 Visual plan improvement	60
Figure 33 Plan generated including long weekends or holidays	61
Figure 34 Plan generated for the start date as holiday or weekend	62
Figure 35 Plan generated for regular working day as start date.....	62
Figure 36 Plan generated for a longer time period.....	63
Figure 37 LK applied to the route with 10 nodes	63
Figure 38 With Lin-Kernighan generated route	64
Figure 39 Best randomly generated route.....	65
Figure 40 With NN generated route	65
Figure 41 Algorithm applied to all cluster data.....	69

List of Tables

Table 1 Example of route nodes and calculation time it takes	13
Table 2 Costs for the first case	24
Table 3 Costs for the second case	25
Table 4 Costs for the third case	26
Table 5 Requirements to time constraints	29
Table 6 Variable tour costs	31
Table 7 Comparison of problem-related approaches	35
Table 8 Comparison of clustering algorithms	40
Table 9 Comparison of DBI values for k number of clusters	44
Table 10 K-Means++ clustering results	45
Table 11 K-Means clustering with custom centroids	45
Table 12 K-Means++ with depots	45
Table 13 Agglomerative clustering results	48
Table 14 Overview of the distribution for each depot	49
Table 15 Case scenario test	52
Table 16 Data structure for depots DataFrame	54
Table 16 Data structure for depots DataFrame	54
Table 17 Best path acquired with random generation	64
Table 18 Comparison of time complexity	66
Table 19 Comparison of route optimality	66
Table 20 Comparison of approaches for a longer route	67
Table 21 Minimal routes found with different approaches	69

1. Introduction

Global supply chains have been imposed to the biggest challenges due to the tragic spread of coronavirus since the end of year 2019. Its impact was immense as it uncovered supply chain vulnerabilities most businesses were unaware of. World spread forced isolation was a necessity, which inevitably led to long term changes: shifting to more flexible and resilient business models. Priority was given to speed of decision-making, greatly facilitated by specialized IT solutions. Business leaders are more than ever responsible for sustainable operations, satisfied customers and wellbeing of their employees. Recent Accenture research has shown that 75 % of fortune 1000 companies experienced a negative to severe impact of COVID-19 on its performance [1]. It also indicates the growing cost of supply chains and operations, which commonly account for the largest part of all costs.

Labor shortages, short-term changes and demand changes are most challenging in the field of OR. The analysis of requirements and current business processes should result in recommendations for consistent and adaptive planning. In our case we see a big potential of improving the way the current scheduling is being planned. This can be achieved after a comprehensive analysis of current processes and comparison of existing successful solutions for the similar problem.

The optimization problem of operational planning and scheduling has been studied since 1930 and first formulated as a TSP, which served as a starting point for further complex problem formulations in various fields of Mathematics, Computer Science etc. The main idea is to define the shortest path, which goes through all the nodes once and terminates in the start node. Since Karp in 1972 has proven the TSP to be NP-hard, the research around it has started to evolve, resulting in new algorithmic approaches. Another problem, considered as a generalization of TSP, is called VRP, and compared to TSP, generates multiple routes.

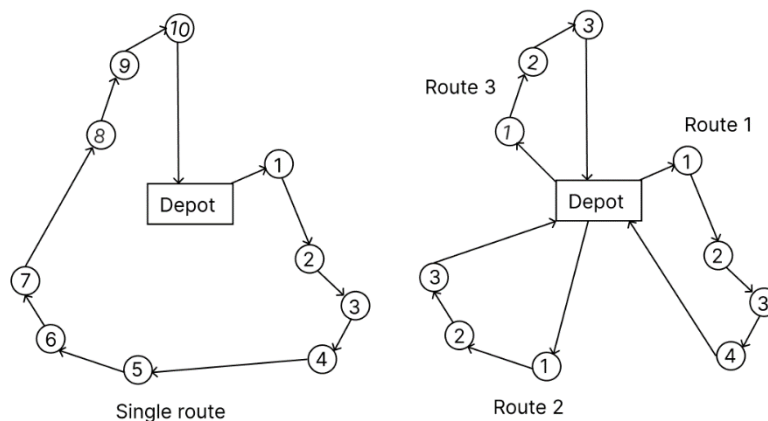


Figure 1 Difference between (left) TSP and (right) VRP

Since the first formal introduction of the VRP in 1959 [2], various approaches to solve it have been introduced to the scientific world, for example VRPTW, VRPPD [3], CVRP. Its classical definition and spawned variations have found its high practical demand in such areas as transportation, logistics, healthcare etc. The classical solution to the problem includes designed routes with the least total cost, start and end depots (location in which the route starts/ends), visiting each node exactly once with the capacitated vehicle of specific type [4]. Those are examples of requirements that are common for such a type of problem, but its complexity grows with additional constraints like delivery times, maximal distance, resource bottlenecks and others.

Various research has been done in order to find a close to optimum solution. It is however not possible due to its NP-hard nature, meaning the number of possible solutions to the VRP is of the

order of $n!$ where n is the number of nodes (locations the vehicle must visit) in the overall graph [5]. If the power of the polynomial gets too large, the problem instance cannot be solved even in average cases or most of the times. Although there is no known polynomial-time algorithm, there is no proof that algorithms for NP-hard problems cannot exist [6].

All approaches can roughly be categorized into classical heuristics (till 1990) and metaheuristics [4]. Some of the most efficient algorithms tend to be the latest heuristics like Tabu Search, genetic algorithms, or Simulated Annealing, which are consistently delivering better results in terms of solution quality and time complexity. Despite worst-case solutions being possible, these algorithms yield quick solutions that are near to optimum or even optimal. With the help of optimization or simulation models, operational planning becomes easier to solve using OR.

As pointed out by W. Domschke et al [7], a designated software ensures the most reliable decision-making. Software products, often used for planning or scheduling, are for example spreadsheet programs such as Excel, commercial simulation or computing platforms, or open-source developer tools which could be adjusted to the business needs. Market offered commercial software can be best described as a framework as they often require additional customization and modification. Each individual business has its own constraints and requirements that reflect actual challenges in production and delivery processes. Every requirement limits either delivery time, resource distribution or capacity etc., which directly increases the hardness of the problem. When planning routes, we obviously want to get the best result possible. The simplest approach, known as a Brute-Force or exhaustive algorithm, is an enumeration of all nodes possible and its check for problem requirements satisfaction. Thus, it always gives the best solution, but even when computed on a software, it can lead to enormous computational costs when the number of nodes is growing. Below is the time complexity directly related to the number of nodes to solve a TSP, which is applied to a single vehicle, visiting multiple destinations or nodes in a single ride.

Number of nodes	Approximate Solution Time
10	3 milli-seconds
20	77 years
25	490 million years
30	$8.4 * 10^{15}$ years

Table 1 Example of route nodes and calculation time it takes

The solution time spikes instantly along with the growing number of route nodes. This can be avoided with either hybrid approaches or the latest metaheuristics and a limited set of most significant costs and other metric functions.

1.1. Goal of the Thesis

This work focuses on the analysis of the algorithms and latest heuristic and metaheuristic algorithms and finding the optimal solution to solve routing vehicle problem on the example of one healthcare company, which provides Germany-wide services.

Contribution to the route construction and optimization requires deep analysis of the problem specific to the company: what requirements and constraints do stakeholders have, what data is required for input and what output is expected, which stages and actors are involved. The work pipeline consists of following stages:

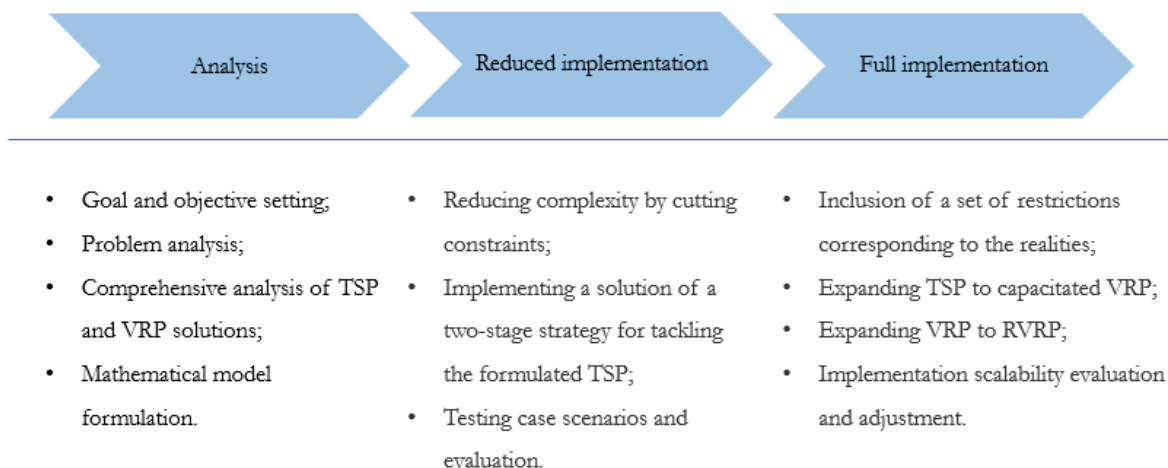


Figure 2 Stages of thesis

Understanding the core problem involves analysis of the current limitations and available resources as well as already existing solutions. At the initial stage it is extremely important not to complicate the conditions of the problem and to proceed from the ideal situation, not including other input parameters that determine the level of stability and realism of the model. All requirements and problem associated conditions will be described in later sections.

The general formulation states that input parameters consist of a set of predefined destinations, located in a coordinate system, and a fleet - a limited number of available identically capacitated vehicles. The goal is to allocate all starting points, so called depots, and to calculate all routes so that all requirements are satisfied, and the overall cost is set to minimum. The result is a constant-factor approximation algorithm for a distance-capacitated VRP.

The first and most crucial step is to correctly limit these parameters which then later affect computation time and overall accuracy. All information provided is legally obtained from the wellabe GmbH and can be used for further research. Goal of the thesis and expected outcomes were defined together with stakeholders. Main priorities for the company are its employees' comfort and the cost minimization. Corporates see the study as a technical support prospect for project managers and foundation for further development and implementation. The knowledge, models and recommendations obtained from the study can be further used for the company's own purposes.

1.2. Research Structure

Daily planning is performed by an operations manager with data retrieved from .xls sheets. The document is daily maintained and updated but is not straightforward when it comes to fast decisions. There is also no clear process on how the trips will be organized, but more often it is managed by the "first came, first served" principle.

In order to build a reliable algorithm to use for real operations planning and satisfy stakeholders' expectations, following steps are needed:

- define the constant set of the input variables - resources and its characteristics: vehicle quantity, vehicle capacity, number of start depots, number of destinations, time constraints, distance constraints, any other requirements etc.;
- considering the specific of this particular case, choose the routing strategy which can maximally cut the overall costs;
- simulate the build routing model;
- optimize model, making hypotheses for changing the input variables;

- run the new simulation and by comparing results, choose the most optimal solution.

The problem is considered as middle-sized with 154 nodes in total. Thus, the solution should be resistant enough in case the number of nodes grows. Ideally, we get an algorithm which will support managers in tour planning. It should provide a schedule for all nodes and use the maximum of resources, not taking any risks into account yet. After literature analysis, we would follow one of the defined guidelines from the latest research done in the similar field. Before starting to work with the data, it should be prepared for the actual analysis. This will be ensured by the fulfillment of a few steps listed below:

- gather relevant data;

In our case the data is currently stored in several Excel spreadsheets, so the retrieved raw data can be further processed in the IDE and turned into a readable format to perform any further preprocessing operations.

- cleanse and format the data;

As documents are maintained by employees, it is not excluded to have typos or missing data. Therefore, validation and profiling are required. After that it should be ready to be organized into complete and accurate data sets.

- combine the data;

Data will be transformed and enriched to become more useful and insightful. It can be appended, categorized to form more holistic datasets.

- analyze the data.

Any change to the dataset should be carefully considered. Chosen algorithms should be compared to get optimal results.

After preparing the data, we want to see if it builds natural groupings or not. As we have more than 150 destinations across all Germany, we want to identify if and what clusters can be formed and assign each of them to start depots. The clustering strategy is later to be defined. It will be the first preprocessing step to simplify the complex routing problem.

With the information obtained this way, the entire service area is divided into individual smaller areas, according to certain criteria defined by the algorithm. A vehicle driver is responsible for one cluster and starts his tour at an optimal point called central depot, which will be marked by the computing algorithm. Within each area, a route must be found for the driver so that the route is kept, if possible, within predefined timeframes and the costs incurred by the company are kept as low as possible. The time of drivers traveling should not exceed the predefined value as well.

This thesis deals with the latter point of the overall project, finding the optimal route in each cluster. An additional complexity could be added due to the condition that drivers, which are company employees, should not work longer than 6 days a week and 8 hours a day, including the time on the road. Sometimes it is impossible though to satisfy demands of all stakeholders, therefore some requirements may be compromised or should be neglected, there is also a chance to find an alternative and change the model respectively.

The following phases arise in the context of this work:

1. Preparing data and defining solution constraints.

By providing an accurate set of constraints specific to the situation, it is possible to screen out most algorithms easily. The aim is to break down the complexity into single, observable steps that can

be iteratively applied to achieve the best possible calculation. To define expected outcomes, stakeholders should set metrics, and the main goal can serve as an objective function in the model.

2. Creation of a mathematical model.

According to the problem described above, a mathematical model is to be formulated that includes all necessary conditions so that it can be used as the basis of the post-delivery control to be solved. In doing so, a model from the general theory of route planning is adapted to the test case.

3. Aligning the model with requirements.

Considering the input set, an algorithm for a CVRP is to be defined to model an initial simplified solution. A higher-level metaheuristic is to be used in case of additional complexity.

4. Programming and parameter analysis.

The developed solution approach is to be implemented and tested with the programming language python. An analysis of the parameters occurring in the algorithms is to be carried out and a recommendation for the most suitable choice of these variables is to be made.

2. Methodology

To understand the difficulties inherent in the VRP, it is important to realize that today instances of the TSP with several thousand locations can be solved exactly using Branch-and-Cut methods, whereas with the VRP, instances with more than 70 nodes are already a major challenge. There are two classes of algorithms that can contribute to the optimization, depending on the specifics of the problem: exact algorithms and modern heuristics. For that reason, we were analyzing time-proved algorithms, as well as the latest heuristics and metaheuristics like Tabu search, genetic algorithms etc., which are proved to be successful in solving both problems.

IP methods provide possibilities to find an exact solution for problems with linear objective function. If the objective function is linear, methods of ILP contribute to the solution of the problem. Such optimization problems of IP and ILP are generally using permutations, sequence, and assignment to determine optimal routes. Also, the decomposition into single problem subsets falls into this task area. Decomposition is a method of breaking down a complex problem into smaller, more manageable subproblems, and solving them individually. This can be useful in solving optimization problems like TSP and VRP because it can reduce the size of the problem and make it easier to solve. For example, in TSP, one common decomposition method is the divide-and-conquer approach, where the problem is divided into smaller subproblems, and the solutions to these subproblems are combined to find a solution to the original problem. Another decomposition approach for TSP is the Lagrangean relaxation, where the problem is relaxed by adding constraints, and then the solution is refined using a heuristic approach.

Similarly, in VRP, decomposition can be used to break down the problem into smaller subproblems, such as finding the optimal route for a single vehicle or finding the best way to allocate customers to vehicles. This can be useful in reducing the complexity of the problem and making it easier to solve using optimization techniques. Overall, decomposition can be a powerful tool in solving TSP and VRP, as it can simplify the problem and make it easier to solve. However, it is important to choose an appropriate decomposition method for a specific problem, as some methods may not be suitable for all situations. These optimization problems can be very complex and computationally intensive, so sophisticated algorithms and techniques are often required to find near-optimal solutions in a reasonable amount of time.

Exact algorithms and heuristics can have varying time complexities, depending on the problem they are solving. Exact algorithms are algorithms that provide a guaranteed optimal solution to a problem. They typically have a higher time complexity than heuristics, and as a result, they are often impractical for larger inputs. Examples of exact algorithms for the VRP and TSP include Branch and Bound, Branch and Cut, and Dynamic Programming. These algorithms work by systematically exploring all possible solutions to the problem and selecting the best one.

Heuristics, on the other hand, are algorithms that aim to provide near-optimal solutions to problems in a more efficient manner than exact algorithms. They typically have a lower time complexity than exact algorithms, but the solutions they provide are not guaranteed to be optimal. The time complexity of heuristics can range from linear to exponential, depending on the problem and the specific heuristic being used. Examples of heuristics for the VRP and TSP include the NN algorithm, the Greedy algorithm, the 2-opt algorithm, and genetic algorithms. These algorithms work by making intuitive decisions or making random mutations to find a good solution to the problem.

Before solving a VRP, we must first understand the TSP. In the last decades TSP heuristics have made remarkable advances in computational time and optimality. Below are definitions of those

widely used algorithms, which will be later considered for the implementation of the algorithm of the stated problem.

1. Minimum Spanning Tree (MST) algorithm - This algorithm starts by sorting all edges in increasing order of their weights and then selecting the smallest edge that does not cause a loop. The MST is used as a preprocessing step for other VRP algorithms, such as the Christofides algorithm.
2. Nearest Neighbor (NN) algorithm - The NN algorithm starts at the depot, selects the nearest node that has not yet been visited, and continues in this way until all nodes have been visited and the tour ends at the starting depot.
3. Greedy Algorithm - This algorithm involves selecting the best option from a subset in the current step, which is not always optimal and can lead to a longer route. Some successful applications of the greedy approach are Dijkstra's algorithm and Huffman coding.
4. Nearest Insertion (NI) algorithm - This algorithm selects the closest node from the starting point, and when selecting the next node, it should be the closest to those already in the selection and inserted so as to cause the smallest increase in length.
5. Farthest Insertion (FI) algorithm - This algorithm selects the farthest node from the starting point and when selecting the next node, it should be the farthest from those already in the selection and inserted so as to cause the smallest increase in length.
6. Christofides Algorithm - This algorithm first finds the minimum spanning tree, then selects the best node that has an odd number of connected nodes and excludes the repeated nodes to form a Hamiltonian cycle.

All of these methods can be used in a combination with a heuristic in order to solve a VRP. The choice will depend on the specific requirements and constraints of the problem. To be able to model a possible solution of any real-world problem, the problem should be simplified and broken down into several steps. When a solution for the core complexity is found, remains stable in tests, and allows flexibility, then an additional complexity can be added.

In this thesis various approaches and algorithms to solve the VRP will be researched to propose a viable implementation solution for distance-constrained routing problem. The minimum final cost of the route must be ensured and serves as a measure of the implementation quality. The solution must always be guaranteed and available in a readable format to the operator. Research shows the advantage of heuristics over classical methods, first of all in the speed. We assume that latest heuristics could deliver best results for the problem introduced in this work.

Tour planning focuses on creating the most cost-effective schedule possible, the so-called tour plan, by concretely assigning each customer to a tour, which can be used to regulate the collection or delivery of certain goods. A tour is identified by a precise order in which the driver serves his customers.

Depending on whether each edge is to be visited at least once or each node (customer), a distinction is made between edge-oriented and node-oriented planning problems. If there are a lot of nodes close to each other, it makes sense to combine them as an edge. An example of this is waste disposal in individual streets. If, on the other hand, there are rather large distances between the customers, these are considered as individual points of the plan to be approached. This thesis deals with such a node-oriented problem. The solution of a route planning problem usually has two aspects: Clustering specifies which destinations can be grouped into a tour. Routing defines in which order customers at each destination are served within a tour. Depending on which aspect is executed first, a distinction is made between performing route calculation first or the clustering. In the first

case, an overall tour is first created that includes all nodes and then this is broken down into individual shorter tours. In the second case, nodes are combined into meaningful clusters and then the optimal sequence of customers is searched for within the clusters. Literature research in a later chapter should result in a clear preference between these two methods.

2.1. On the Tour Planning Problem

The tour planning problem is one of the most researched topics in operational research due to its relevance especially in logistics. To illustrate the situation, let's imagine the following set of given vehicles (V), nodes (N), cost of the associated route from depot arc (c), resource demands of each node (d). Here each start depot is assigned to a vehicle, that is why we mark those with V .

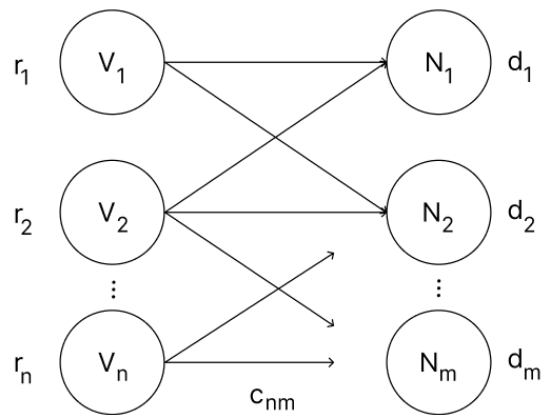


Figure 3 VRP basic scheme

Even with the latest methods, we cannot guarantee that the solution found would have the best outcomes possible. Multiple depots or route duration restrictions already make this problem NP-hard. To simplify the problem, we construct routes scheduled in advance, not including in-time or dynamic optimization. The complexity can be as well minimized by planning for the short term and for a smaller number of customers, which should be served during this time period.

2.2. Route Planning for Multiple Start Depots and Capacitated Resources

Operational research literature confirms the importance of the initial planning procedure. The current routing planning is suboptimal as it often includes unexpected, short-term changes, which leads to inconsistency and misleading information. In general, it is worth considering automated approaches such as heuristics, mathematical programming, or metaheuristics, especially for large-scale VRP problems. These approaches can provide more efficient and reliable solutions and can help to save time and reduce the risk of human errors. However, the choice of method depends on the specific requirements of the problem and the available resources.

The usual planning process follows below steps:

1. Customer initiates a call and asks about the availability of service on specific dates. Each customer has an agreement on a certain number of service units (from here “check-up”) he can book in the service provider company.
2. One customer asks for N numbers of check-ups for N number of employees, so that one check-up is performed with an individual employee.
3. In one day, a maximum of 16 check-ups can be performed. Thus, with a minimum one working day, but no more than 6 days of providing a service in the customer location.
4. Each customer has its own location, where the service should be provided.

5. The number of provided check-ups (maximum 16 per day) depends on the customer, if more is desired, days of service on the customer site are extended.
6. Customer and service provider negotiate the dates and number of service days.
7. After the agreement is met, the service provider starts to plan the resources amount required for a service provision.

Each point in more detail:

- The B2B customer (from here - customer) is the initiator of the call. This can be an existing customer, or a customer interested through referral marketing. The call is made without a specific rule. Following types are possible:
 - The customer is not new, and the location is already in the DB / list;
 - The customer is not new, and the location is not in the DB / list.
- The customer names a location that can provide necessary rooms, parking space, etc.;
- Depending on the number of check-ups, a compromise for the number of days on the customer site is found;
- The service provider can either reject or accept the proposal. Nevertheless, he always tries to take the customer's wishes into account;
- The service provider can either give initial approval (and make an agreement at the second contact) or cancel if there is not enough time or resources for planning before the assignment;
- The priority in planning generally remains the profit from services offered - this should cover the costs;
- As resources: workforce (coaches), equipment (software, materials, etc.), time (enough time between the assignments themselves).

Factors that can hinder the route planning and are not included in the formulation of the current problem, as they drastically increase the overall complexity:

- Missing data
 - Number of checkup days are not / cannot be pre-planned for all customers;
- Sudden events
 - External factors
 - customer cancels checkup at short notice;
 - customer wants a checkup, but the time does not fit;
 - Internal factors
 - Employee / driver (coach) is absent due to illness (before or during the assignment);
 - Car is out of order;
 - Equipment (hardware) is out of order;
 - Materials are out;

- Lack of resources
 - There are not enough resources (the time period is already occupied, manpower occupied, etc.).

Further restrictions can be applied in order to get a more realistic optimization model:

- The duration of the mission should be a maximum of 5-6 days (time limit);
- The journey in one direction should take a maximum of 2 hours (distance limit);
- An employee (“driver”) cannot overnight at the hotel on the weekends, but must drive back to the start depot and back to the assigned customer;
- It is extremely useful to have a "backup" employee in case of an emergency.

2.3. Complexity Analysis

The goal of this work is to explore the discipline of OR in order to develop mathematical and then computational methods to the optimization routing problem of the company providing preventive healthcare services. Proposed implementation approach should show the prospect and benefit of using modern heuristics and automation versus manual intuitive planning, which can be bad because it may not always result in the optimal solution and may be subject to human biases and limitations. Additionally, it can be time-consuming and error-prone, especially for large-scale VRP problems. An underlying RVRP cannot be solved in the scope of this work as it entails extra constraints and objectives that make it more complex and challenging to solve. In a Rich VRP, the vehicle routes must not only minimize the total distance traveled, but also consider a variety of additional constraints and objectives, such as time windows, capacities, delivery priorities, and multiple depots. This leads to a more intricate optimization problem, which requires the use of sophisticated algorithms and heuristics to solve efficiently.

Therefore, an expected outcome of this work is to develop both full RVRP and relaxed DCVRP mathematical models. The latter doesn't consider the capacity constraint of the vehicle or the fact that customers may be served from multiple depots. In addition, it is mandatory to visit all destinations and end at the depot unless the maximum permissible length of the route or its duration is exceeded. The problem is to find the optimal routes for the vehicles that minimize the total distance traveled, subject to the distance constraint for each vehicle. There are numerous approaches that were developed in the last fifty years, so we need to have clear expectations on what we want to achieve and which limitations a particular VRP has before solving it. Classification distinguishes exact approaches like Branch and Bound or Branch and Cut algorithms and heuristic algorithms. For problems containing more than 30 nodes, heuristic is a more reasonable choice in finding a solution.

Helena R. Lourenço et al. outlines the importance of the metaheuristic simplicity, both in design and practice [8]. Ideally, we get an effective algorithm, which solves generalized routing problems independently from any additional specifics. There are several approaches to simplify a MDVRP, including the following:

- Aggregation: One approach is to aggregate the customers into clusters based on their proximity to each other, and then treat each cluster as a single customer. This simplifies the problem by reducing the number of customers to be serviced, and also makes it easier to solve the problem computationally.
- Fixed routes: Another approach is to use a fixed route structure, where each depot is assigned a specific set of customers to serve. This simplifies the problem by reducing the

number of routes that need to be considered, making it easier to solve the problem computationally.

- Relaxation of constraints: Another approach to simplify the MDVRP is to relax some of the constraints, such as the maximum vehicle capacity or the maximum distance that a vehicle can travel. Relaxing these constraints can make the problem easier to solve but may result in sub-optimal solutions.
- Heuristics: Heuristics are approximate algorithms that can be used to find near-optimal solutions to the MDVRP. They can be used to simplify the problem by providing a fast and efficient solution, even if the solution is not guaranteed to be optimal.
- Approximation algorithms: Another approach is to use approximation algorithms, which can provide a guaranteed bound on the quality of the solution. Approximation algorithms can be used to simplify the problem by providing a fast and efficient solution that is close to optimal.

Relaxation of constraints is what will be achieved with the formulation of the relaxed mathematical model in a later section.

Capacitated VRP with time windows, which is referred to as a multi-depot VRP and the location routing problem, was successfully solved by Laporte et al. [9]. Fermín Cueto et al present their own terminology for the term “trip” to address a multi-trip problem, where a vehicle drives from the depot to the customer and back. The term route they use to describe a single path the vehicle follows during a trip [10].

The initial step will be to find optimal routes for a single depot. The goal is to optimize route planning for given start-depots and destinations, distributed across Germany. With the implemented model, we would like to get an optimized routing plan for a given period of time and consequently achieve a cost reduction of feasible routes.

To handle big-sized problems, they are often solved in multiple sequential steps to make them more feasible and less constrained. The principle of cluster-first route-second is widely used for solving VRP and is proposed as a solution for reducing complexity of the assignment problem. In the following sections a stated VRP will be broken down to local TSP problems.

Below we will define mathematical formulations for both RVRP and its simplified version – DCVRP. A full mathematical model is a complete formulation of the problem, including all of the constraints and objective function. Full mathematical models are typically more complex and harder to solve, but they provide a more accurate representation of the problem and can lead to better solutions.

A relaxed mathematical model, on the other hand, is a simplified formulation of the problem that relaxes some of the constraints or makes approximations in the objective function. Relaxed mathematical models are typically easier to solve, but the solutions may not be as accurate or optimal as those from full mathematical models.

It is necessary to define the objective function, the constraints, and the variables as both problems are optimization problems. In the VRP, the objective function typically expresses the total cost of the routes, which can be a combination of the distance traveled, the fuel consumed, the time spent, or some other measure of the cost of the routes. The constraints typically denote the limits on the resources available to the vehicles, such as the maximum capacity, the maximum distance that can be traveled, or the maximum time that can be spent. The variables represent the decisions that need to be made, such as the order in which the customers are visited, the routes taken by the vehicles, and the allocation of customers to vehicles.

In the TSP, the objective function typically characterizes the total distance traveled by the salesman, and the constraints represent the limits on the travels, such as the maximum distance that can be traveled or the maximum time that can be spent. The variables affect the order in which the cities are visited.

A well-defined mathematical formulation is essential to the solution of the VRP and TSP, as it provides a precise and unambiguous definition of the problem. This allows for the development of algorithms that can be used to find optimal or near-optimal solutions to the problems.

There are a few cases, which can be differentiated by the number of customers to be visited in one trip and service days to be performed for a single customer. The minimum number of days spent at the customer site - one, the maximum depends on the units of service, which are concluded under the contract and the number of working hours of the employee providing the service. All travels can be divided into three types: routes with short distances, routes that include long distances and require hotel overnight and traveling from one node to another. Below described cases include real estimated costs but a few main cost factors are important to consider when running an optimization algorithm. A few parameters will be added for the total cost calculation, for example a fuel price, which has a direct impact on the overall route cost. These cases describe a travel for one vehicle, which serves its allocated customers.

2.3.1. First Case

A route, consisting of short paths and beginning at the start depot, continues to a single node, which stands for a customer, and returns back to the depot on the same day. The travel distance between nodes is not exceeding 2 hours. In addition, we have restrictions on the amount of working hours, which should not exceed ten hours. This condition is a great factor, affecting the employees' satisfaction. Thus, we would like to be able to find the most optimized route with shortest paths.

Below are costs, that represent a typical one-day case:

- fuel spent for the ride - 80 euro;
- employees' wage – 330 euro;
- overall expenses included in the service maintenance - 237 euro.

Total cost for the use-case is 647 euros for 8,3 working hours, including time spent on the road and on providing service on customer site.

Case 1

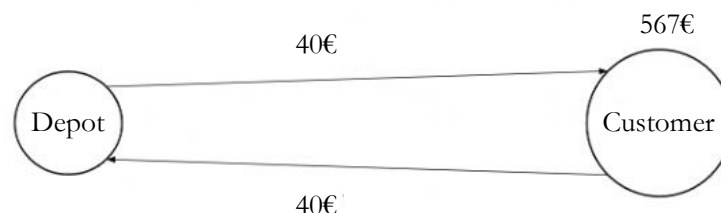


Figure 4 Trip for the first case

Below are fixed costs, associated with the case, regardless of the number of check-ups provided in one day or time spent on the site. They can be used for modeling a similar scenario. Information on expenditures has been estimated and provided by the company, therefore cannot be described in detail.

Name of expense	Amount in euro
Materials	125
Fuel	80
Starting or finishing preparations	45
Amortization	67

Table 2 Costs for the first case

By going to a new node, the employee (the driver) is responsible for preparing the room and equipment before starting the working day, which requires both time and costs each time it is a new location. Preparation and finishing costs are made only once in the first and the last day of performing a service. Therefore, if a customer is served for a few days in a row and the vehicle doesn't have to be returned to the start-depot, then these costs would be counted only once.

This case is possible if some of below statements are true:

- only one day of service is to be performed on the customer site;
- the total time needed to reach the destination and return back to the depot takes less than 2 hours;
- hotel overnight is more expensive than the travel cost.

2.3.2. Second Case

The second case includes unoptimized or just longer routes, which last 2 hours or more. As there are 7 depots, from which a vehicle drives to the customer, and 154 customers total located in Germany, the distance between routes varies. In our interest is to look for shortest possible routes, which reduce the transportation cost and are more beneficial to employees' health.

Case 2

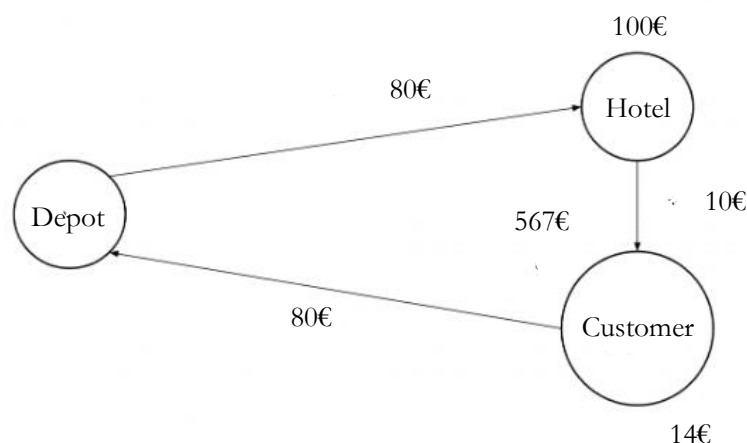


Figure 5 Trip for the second case

In this case, the driver starts his way from the depot, but as it takes 2 hours and the working day starts at eight o'clock in the morning, he would have to come to the destination the day before to overnight at the hotel. Transportation from the hotel to the customer is a default value set to ten euros in the scope of this work. Working time increases up to 9,83 hours. Wages correspond to

previous first case – 330 euro. The daily provision cost is included as well. All costs presented in the table below are constant, so the main goal is to affect traveling expenses.

Name of expense	Amount in euro
Preparation for service on site	45
Finishing preparations	45
Amortization	67
Hotel overnight	100
Provision	20

Table 3 Costs for the second case

An optimization that could be suggested here is as follows: can customer nodes be evenly distributed between depots, so that a single ride in one direction won't take more than 2 hours? A possible assumption would be to form clusters, where each of them is served by a single depot. What rules determine if a node should be visited, and the route stays optimal. Total costs for one customer served in this use-case: 851 euros, which accounts for 33% of the profit.

2.3.3. Third Case

The following scheme shows costs for four days service which is a full working week or forty working hours. This case illustrates visiting four customers, one each day. This case is the most expensive as visiting a new customer is associated with travel costs, preparations before and after service and employee maintenance budget. It also requires extra time for software preparations and cleanup afterwards. Additional costs can be avoided if all services for a particular customer can be provided at once, as it won't cause extra installation costs.

Here the main focus is on minimizing total travel time thus finding shortest routes between nodes. Staying at the hotel should be more profitable than going back to the depot, but we should also take the employee's wellbeing into account and do not allow working overtime.

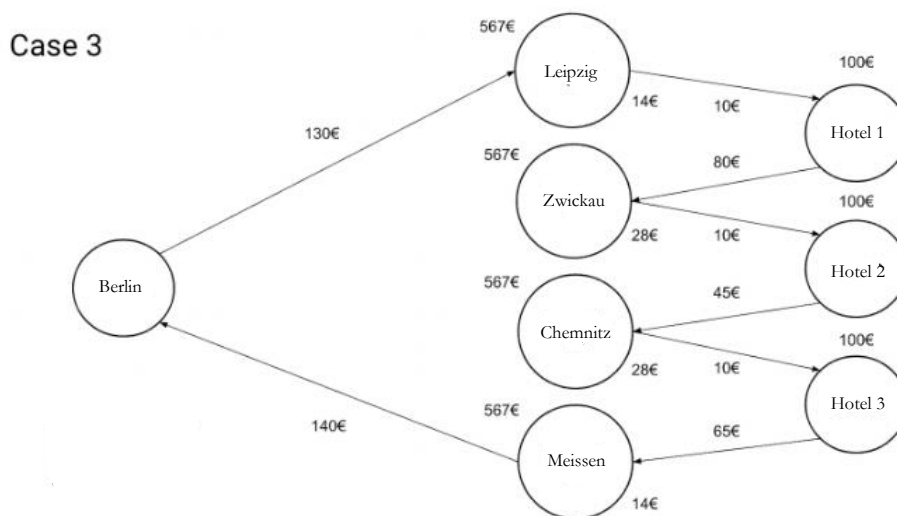


Figure 6 Trip for the third case

Below costs will apply each day as the service will be provided to the new customer.

Name of expense	Amount in euro
Preparation for service on site	45
Finishing preparations	45
Amortization	67

Table 4 Costs for the third case

The implementation should be able to cover all the above use-cases and provide an output in a form of the plan file, readable for the operations manager. The proposed solution must be beneficial and more advantageous to use than former planning methods and tools. Its main criterion must have a direct positive effect on the total cost.

From the above problem description, it is obvious, that there is a strong need for optimization that directly affects the profitability growth, which can be calculated as:

$$\text{Contribution margin} = \text{Revenue} - \text{Variable Costs}$$

To avoid loss in contribution margin, variable costs must be optimized and/or the sales price adjusted.

2.4. VRP Model

Below mathematical model serves to describe an objective function of the asymmetrical RVRP. In an Asymmetric VRP, the cost of traveling from one customer to another can be different in different directions, i.e., the cost of traveling from A to B can be different from the cost of traveling from B to A. Symmetric VRP is generally easier to solve compared to the asymmetric version, as the symmetry of the problem leads to simpler mathematical models and algorithms. However, in real-world applications, it is often the case that the cost of traveling between 2 points is asymmetric, so the asymmetric version of the problem is more appropriate to model the problem. As for the objective of RVRP, it is defined as a combination of goals such as finding the maximum profit, keeping customers and employees satisfied subject to constraints such as the capacity of the vehicles, the time window for customer visits, and the maximum distance that each vehicle can travel.

Linear programming formulation is widely used in the OP to obtain an optimal solution for a given set of limitations. A fleet of V homogeneous vehicles, each located at the depot d . Each customer q has a non-negative demand expressed by the number of service days $S(q)$ allocated to this customer. Each customer is serviced by a single vehicle within its capacity. Each route must start at the depot and end at the same depot. The following mathematical formulation has an objective of defining a set of vehicle routes with minimal total travelling costs. Below notation expresses our constraints in terms of decision variables.

Sets:

- $D = \{d_1, d_2, \dots, d_n\}$ for a depot set, represented by its geolocations;
- $V = \{v_1, v_2, \dots, v_n\}$ for a vehicle set of equal capacity;
- $Q = \{q_1, q_2, \dots, q_n\}$ for a set, represented by customer geolocations.

Cost variables:

- C_m for fixed material costs;
- C_L for hardware maintenance costs;

- C_p for variable service preparation costs;
- C_f for fixed fuel cost per km;
- C_w for an hourly wage of the employee;
- C_a for accommodation costs;
- C_n for provision costs.

Variable costs associated with service:

- production supplies – materials needed to provide a service;
- freight in/out costs – transportation expense incurred for the service;
- hardware operating costs – expenses for the hardware maintenance before and after the service was performed.

Parameters:

- U represents a number of performed service units (so called “check-up” must be taken as a single unit);
- $T(e)$ represents a time or workload available for the particular employee (driver) on a working day d ;
- $S(q)$ represents a number of service days, booked by the particular customer q ;
- k_{ij} represents a distance in km between customer i and j or the edge ij – here is more abstract notation of customer nodes is used to represent an edge between 2 vertices;
- k_{dq} represents a distance in km between depot and the customer considering the direction of the route;
- $T(k)$ is a time equivalent to the distance k ;
- x_{vij} represents a binary variable, where 1 means the vehicle v starts its route at node i and ends at node j , otherwise set to 0;
- y represents a binary variable, where 1 is set for the working day, otherwise is a holiday or weekend then set to 0.

The model's objective is to minimize total cost, which is the sum of total traveling cost from depot to all customers and the total fixed cost of the vehicle. Total cost mathematically expressed in the equation below. Thus, the objective can be formulated as follows:

Minimize the total cost of the service provided to customers in a given time period while using all the allocated resources:

$$\sum_{e=1}^{S(q)} C_p C_w k_{ij} x_{vij}, i \neq j \in Q \quad (2.1)$$

Subject to

$$\sum_{j=1}^n x_{vij} = 1 \quad (i = 1, \dots, n' \forall n \in Q) \quad (2.2)$$

and

$$\sum_{i=1}^n x_{vij} = 1 \quad (j = 1, \dots, n' \forall n \in Q) \quad (2.3)$$

Here we specify the assignment of vehicles and employees to the route:

$$f(v, i, j) = \begin{cases} 1 & \text{if vehicle } v \text{ travels from } i \text{ to } j, \\ 0 & \text{if otherwise} \end{cases}$$

$$\forall i, j \in Q, v \in V \quad (2.4)$$

An equation of fixed costs for a working day according to the Case 1, excluding transportation costs:

$$C_L C_p C_m \sum_{u=1}^{T(e)} C_w \quad (2.5)$$

An equation of fixed costs for a working day according to the Case 2 or Case 3, excluding transportation costs:

$$C_L C_p C_m C_a C_n \sum_{u=1}^{T(e)} C_w \quad (2.6)$$

Further we specify additional constraints applied to the case described in the Figure 7 below. If the route implies more than one service days, for example $S_q = 4$ for the customer q_1 , at the end of each day it should be checked whether going back to the depot is more profitable than staying in the hotel at the same location. On the one hand, there are costs associated with staying at the hotel, on the other hand going back and forth is causing extra travelling costs. Arriving and leaving the node is associated with service preparations which require a fixed amount of time, which is expressed through a variable prep. In the real case scenario multiple factors such as in advance booked hotels, sudden cancellations (hotel, customer unavailability), emergency with software or employees etc. can affect this decision and are impossible to predict or fully consider in order to construct the best route or reduce expenses at most.

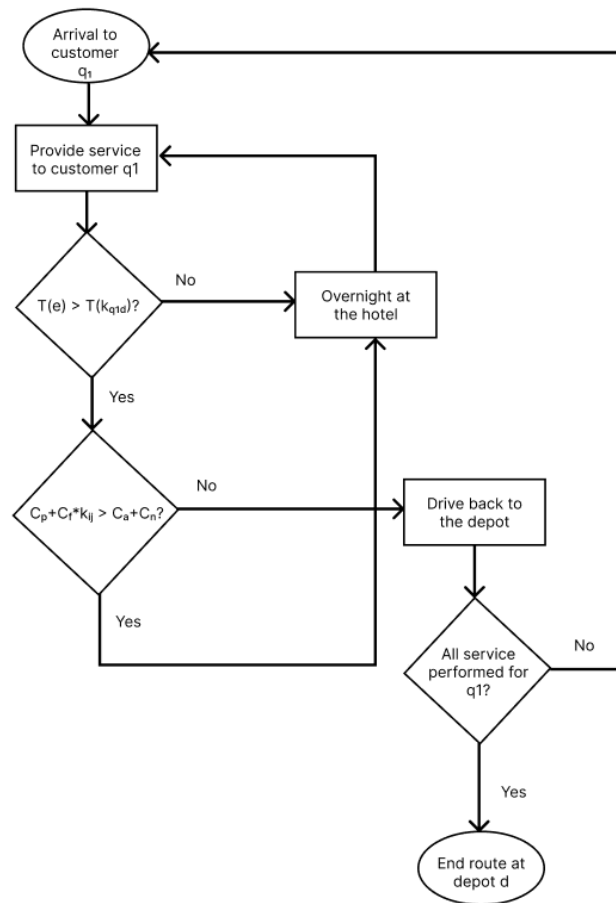


Figure 7 Case with factors affecting the routes' feasibility

To minimize unproductive time gaps, a decision is made on each working day whether to continue the route to the depot or not, and whether going back to the depot is more profitable than driving straight to the next customer location, as explained in the figure above. This decision is strongly dependent on the remaining working time of the employee on that day. The employee's well-being is a high-priority decision variable that should be taken into account first.

Certain costs associated with time are shown in the table below:

Requirement	Duration
One check-up	0,5 hours
Working day	10 hours
Maximum daily time in travel	2 hours in one direction
Maximum working days in a week	5 days
Maximum weekly workload	50 hours
Preparation before service	1 hour
Preparation after service	0,5 hours

Table 5 Requirements to time constraints

Below are some additional constraints, which impose constant conditions on variables V, C:

- the speed of the vehicle used in calculations depends on the road, weather, and other relevant conditions;
- each start-depot can be serviced with different vehicles;
- returning back to the depot depends on the employee workload capacity;
- customer can be visited more than once, if the service cannot be fully performed, depending on the employee workload capacity;
- one vehicle is assigned to one driver;
- if a driver is not able to serve customers, reassign its vehicle to another driver;
- the transportation cost of each vehicle depends on the traveled distance;
- vehicle fixed costs such as amortization are included in the equation;
- time window limitations are defined according to each customer demand;
- in the real-world distance between 2 nodes is often asymmetric;
- the number of vehicles, depots and customers are given but can dynamically change:
 - initial number of vehicles: 7;
 - initial number of start depots: 7;
 - initial number of hardware equipment needed to serve a customer: 7;
 - initial number of total customer locations: 154.

Time windows, that are also present for the stakeholder:

- time windows are proposed by customers and compromised together with the service provider;
- the period of time during which a service can be started, e.g., accept service hours only from 8 to 10 o'clock;
- duration of each customer visit equals to the booked service units;
- work break should also be included and planned;
- non-productive time windows not related to work should be minimized.

As VRP with time windows is hard to solve when containing anything bigger than 100-150 nodes, we have decided not to include them into the current problem definition.

In the table below we define some variable costs which are used in some of the scenarios outlined later. The total cost per km is reduced to a fixed constant variable per km, excluding the maintenance cost. This is set for the vehicle type Caddy with an average consumption of 7 liters / 100 km of fuel "Super" [11]. Cost accuracy can be later enhanced by obtaining real-time fuel prices, for example from the website "Tankerkönig" with the data from more than 14000 gas stations, yet this is out of scope for the stated problem.

Name of expense	Price in euro
Fuel per km	0.2
Daily provision	20
Daily accommodation	100

Table 6 Variable tour costs

One full day, which is considered to be the least costly, has a certain number of service units, each of which lasts half an hour. In total, a worker must complete 16 units in 8 hours, which leaves 2 hours for the transportation to the customer and back, if needed. Though it cannot be guaranteed that there is a solution that satisfies these time constraints, there is a chance to find the most optimal solution possible or at least the most satisfying. The quality of the solution found will be measured by the total number of distances traveled in km, as well as by the time required for the calculation of the route. The main criterion remains the total cost for the overall trip.

2.5. Model Relaxation

This model is a relaxation of a previously formulated model. The objective stays the same while eliminating constraints in order to make the problem simpler to solve. Redundant parameters were taken out as well.

Sets:

- S for total service days;
- Q for node set.

Cost variables:

- C_f for fixed fuel cost per km;
- C_w for an hourly wage of the “driver”;

Parameters:

- U for number of performed service units (so called “check-up” must be taken as a single unit) is set to a constant of 16 and considered to be performed during one full working day S_q ;
- D for one depot for which a route will be constructed;
- S_q for a full working day servicing a customer q from the set of Q .
- $V(d_n)$ for one vehicle assigned to a depot $d \in D$;
- $T(e)$ represents a time or workload available for the particular employee (driver) on a working day d , here set to a constant of 10 hours. Time limitations will not be considered in the model;
- $S(q)$ represents a number of service days, booked by the particular customer q and is set to one day;
- k_{ij} represents a distance in km between customer i and j or the edge ij – here is more abstract notation of customer nodes is used to represent an edge between 2 vertices;

- k_{dq} represents a distance in km between depot and the customer not considering the direction of the route;
- $T(k)$ is a time equivalent to the distance k ;
- x_{vij} is a binary variable, where 1 means the vehicle v starts its route at node i and ends at node j , otherwise set to 0;
- y is a binary variable, where 1 is set for the working day, otherwise is a holiday or weekend then set to 0.

Minimize the total cost of the service provided in a given time period, at the same time using all the allocated resources:

$$\sum_{e=1}^{Sq} C_f C_w d_{ij} x_{vij}, i \neq j \in Q \quad (2.7)$$

Subject to the formulations (2.2, 2.3). The assignment of vehicles and employees to the route according to (2.4).

And a cost equation for a working day, excluding transportation costs is equal to equation (2.5).

Below relaxations of constraints should provide more flexibility to the optimization problem:

- the speed of the vehicle used in calculations is constant;
- each start-depot is assigned to one vehicle;
- customer can be visited only once;
- one vehicle is assigned to one driver;
- time window limitations are not taken into account;
- distance between 2 nodes is considered symmetric;
- the number of vehicles, depots and customers are predefined and remain unchanged:
 - number of vehicles: 7;
 - number of start depots: 7;
 - number of software needed for serving a customer: 7;
 - number of customers total: 154.

The next section includes a comparison of heuristics for deciding on an appropriate strategy and specific algorithms to achieve optimization for the formulated problem.

2.6. Heuristics for the Optimisation Problem

Most real-world problems, that require optimization, cannot be solved with a single method or strategy and they often have a huge number of input variables involved. As the problem scale is large, because nodes are distributed all over Germany, more sophisticated routes are needed. Among them are multiple routes as well, which implies solving a VRP, either classical formulation or its variations [12]. The challenges of the VRP have gained a lot of attention from academics due to its variety of dependent variables such as availability of vehicles and its capacity, multiple trips, pickups and deliveries, time-windows and often the combination of those. Solution approaches

can be roughly divided into three categories: mathematical programming, heuristic methods, and metaheuristics - heuristics, which include MILP models. In this thesis, we consider the variant of the VRP known as multi-depot VRP in which more than one depot is considered (see Fig. 2). The reader must note that most exact algorithms for solving the classical VRP model are difficult to be adapted for solving the MDVRP.

Related research starting from 2010 till today has shown a strong increase for addressing environmental issues, including e-vehicle utilization, CO₂ emissions' reduction etc. Sustainability is our major concern today, which can and should be supported through vehicle route optimization. Qian et al. used real London traffic jam data, consisting of 60 nodes, one depot and one vehicle. He solved this problem by using a column generation based tabu search algorithm.

Contardo et al. used the column generation approach strengthened with the so-called strong capacity constraints and clique inequalities. They do not consider the inclusion of the route length constraint and so experiments are conducted only on those instances without such a requirement. He proposed the solution for a multi-depot VRP (MDVRP) by losing single period VRP, where depots can be modeled as multiple periods. Therefore, any algorithm that solves the PVRP can also solve the MDVRP [13]. This leads to an idea of solving the multi-depot problem with single capacitated VRP. The subproblem is solved by finding the shortest path - an algorithm is going through all possible subsets and marks the route with the least cost and within given capacity bounds. We can simplify our model by making all vehicles homogeneous and not taking its capacity into account, paying attention to the vehicle availability first.

Ramos, Gomez and Barbosa-Póvoa solve the multi-product and MDVRP with an additional limit on route duration. They propose a heuristic, hybrid approach for a medium size problem, with 50 nodes and 3 depots. Optimal optimization results were obtained with the MIP model with the condition that certain restrictions are removed, not violating variables given [14].

Bulhões et al. address the VRP-SL, introducing a pair of an ILP code solution and a branch-and-price algorithm, suitable for small- and medium-scale sized-problems [15]. It can be successfully applied to a profitable CVRP with a set of private fleets, operated by a single driver.

Laporte et al. found an optimal solution for routes not exceeding a specified distance upper bound, in which they formulated an integer program with a constraint relaxation procedure [16].

Dufay, Mathieu and Zhou prove the distance tree can be reduced with the bounded space online bin packing problem [17]. Meanwhile, there are various ways to extend the bin-packing model to be more general in terms of cost and loading.

Garside and Laili propose a two-phased approach, achieved by the integer programming implementation. The heuristic includes a Cluster-First Route-Second strategy, in which clustering is applied, followed by the route building, with regard to existing constraints [18].

The same strategy was used by Choudhari, Ekbote and Chaudhuri: for the clustering method they performed a DBSCAN algorithm and Christofides algorithm for finding an optimal path for each cluster [19]. Below is the comparison table that sums up the approaches used in the last decades for solving VRP with constraints.

Autor	Constraints				Goal	Implemented method
	nodes	time windows	distance	vehicles		
Meng et al. [20]	11	+		+	Minimization of fuel consumption and emissions	Two-phase optimization, fuzzy clustering
Shi et al. [21]	22		+	+	Minimization of the total transportation distance	Closest distance search and sector combination optimization
Gong et al. [22]	18	+	+		Minimization of the number of vehicle routes	Particle swarm optimization (PSO) algorithm
Ramos et al. [14]	50	+		+	Finding local optima	Mixed-integer linear programming models
Geetha et al. [23]	10		+	+	Minimization of the total travel length along with route and load balance among the depots and vehicles	Clustering first and route second methodology
Ho et al. [24]	14	+			Minimization of total routing cost	Combination of two hybrid genetic algorithms
Choudhary et al. [19]	10		+		Minimize traveling distance and/or time among customers	Cluster first and route second algorithm, integer linear programming.
Qian et al. [25]	60		+	+	Minimization of fuel consumption	Column generation based tabu search algorithm
Islam et al. [26]	138			+	Finding local optima.	Particle swarm optimization (PSO) and variable neighbourhood search (VNS)

Autor	Constraints				Goal	Implemented method
	nodes	time windows	distance	vehicles		
Cassettari et al. [27]	>100	+	+	+	Maximizing transportation costs and maximize the workload according to customer demand	Algorithm based on Saving Algorithm Heuristic and 2-Opt

Table 7 Comparison of problem-related approaches

As the solution time proportionally increases with the size of the problem, finding all $n!$ possible solutions, where n is the number of nodes, is not optimal for a big enough n . It will result in a long computation time, including infeasible routes as well or a worst-case scenario. The algorithm must remain stable even with a growing number of nodes. Optimal computational time should also be guaranteed. Most scientists prove that complexity of the problem should be minimized to the most possible extent in order to get insightful results. It can be achieved by elimination of certain rules or requirements related to time windows, vehicle and load capacity, type of fleet, which tend to create complex code structure. The set of requirements can be supplemented after the implementation solves the core routing problem. [28] claim that the two-stage strategy helped them to “reduce the overall complexity for the optimization mechanisms by dividing the solution space into two individual problems”. In our case we would break the problem down into cluster problems, where we get separate VRPs, which can be solved with one and the same strategy. The cluster first and route second (CFRS) algorithm strategy is a popular approach for solving the VRP, including the multi-depot VRP. We found that the above-mentioned researchers, who used this particular strategy, have achieved positive results in minimizing the objective function, which represents the performance measure that the algorithm is trying to optimize. As the objective function defined in this thesis is equal to ones compared in Table 7, we can consider for example integer linear programming or local search heuristics like 2-Opt suitable for route construction.

2.7. Clustering Approach

The decision in favor of clustering was taken because of the widely used and successfully established cluster-first route-second approach, first addressed by [29] and used for solving VRP by [30] and [31]. This technique successfully uses the clustering method to simplify the assignment problem, that is dividing customers into different clusters. In the second step it is practical to use weights or similar selection criteria for the path construction and selecting the best one. Due to the routing problems’ nature, most approaches are designed according to this concept. First phase of clustering is depicted by a simple example below.

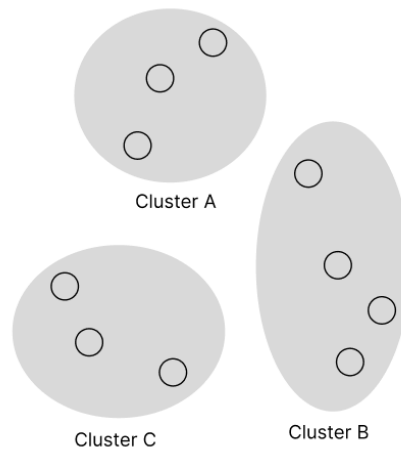


Figure 8 Clustering phase

CVRP or in this case distance constrained VRP is classically solved in this order, where the partitioning is determined first and can be performed by a MILP solution. Route sequencing is then solved for each cluster by any suitable TSP algorithm, for example Lin-Kernighan or Sweep Algorithm. The cluster formation should be flexible to allow adding new destinations, as business is expected to have a continuous growth and should be scalable. The route construction is illustrated for the previously used example below.

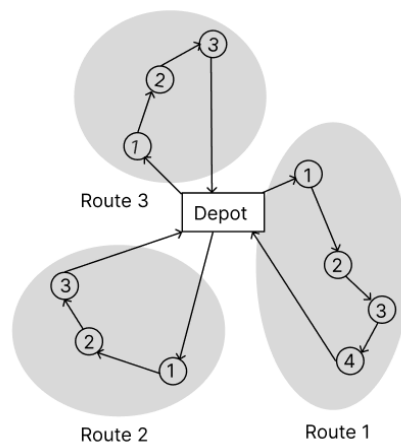


Figure 9 Routing phase

Similar approach implemented by Du and He [32] consists of 2 stages, where after clustering with the traditional k-Nearest Neighbor Search algorithm follows the Tabu Search algorithm, which defines optimal intra-routes.

In conclusion, the algorithm for the formulated problem will be split into 2 successive stages. The first stage serves for data refinement and clustering, to attempt solving a complex multi-depot VRP as a single depot VRP, second stage involves building and optimizing routes. We define the problem stated in this work as a middle size VRP problem, including 154 nodes. A predefined condition prescribes 7 depots, thus 7 centroids (not marked in blue on the image below) that serve all other nodes, representing customers, which are marked blue on the image below.

The first step to reduce the complexity of the problem would be to look at the distribution of the network, as noted in work of Montoya-Torres et al., a VRP with multiple depots can be solved as a multiple individual single depot VRP. For that a requirement of evident nodes clustering around each centroid should be fulfilled. Alternatively, nodes should be served from available depots using available vehicles [33].

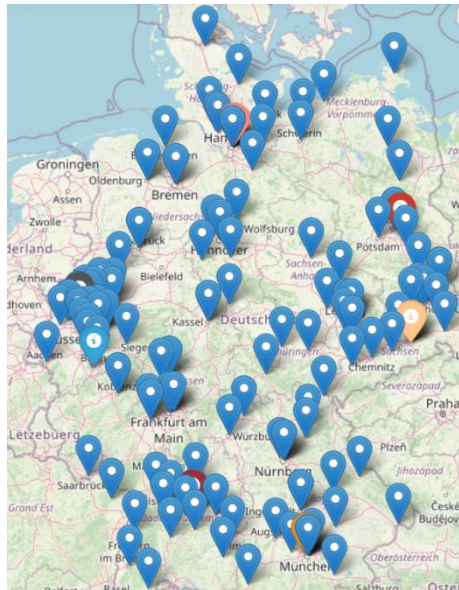


Figure 10 Customers' distribution

We want to divide all nodes and assign them to a particular centroid, so that clusters can be evenly formed. A related question we would like to answer in this thesis: what is the best strategy to form these clusters that ensure a successful route optimization algorithm?

In this section we will create clusters of customer locations, which will be assigned to particular depots, shown on the picture below. Before clustering an additional data, manipulation is required: based on provided geographical data (latitude and longitude) and using an open-source geocoding software with OpenStreetMaps data, exact address information like city or street name can be obtained, added and used later.

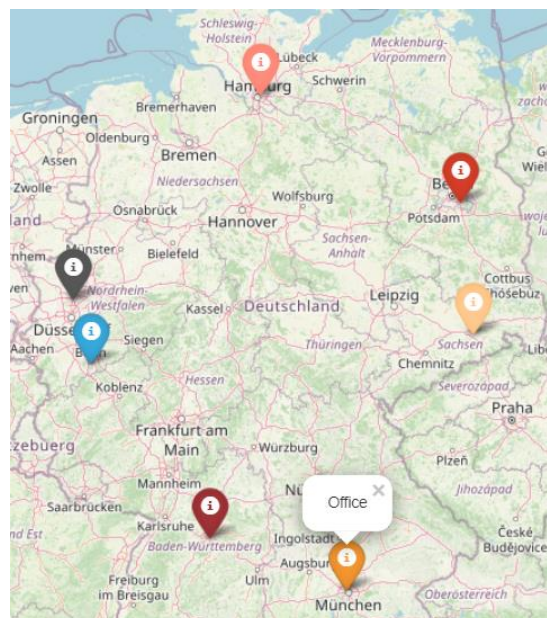


Figure 11 Alternative depot distribution

The process of data preparation and clustering starts from the clean and formatted data saved in xlsx file. This data will then serve as an input for the function, performing a reverse geocoding, that is finding a location name and its address. These addresses will be then accessible for the operations manager or operator to track the scheduling. A distance matrix will be formed for each cluster, as it is required by the algorithm to calculate and compare weights of a pair of nodes. The

cluster will be formed when following the assignment rules all nodes get a label, associated with the particular cluster.

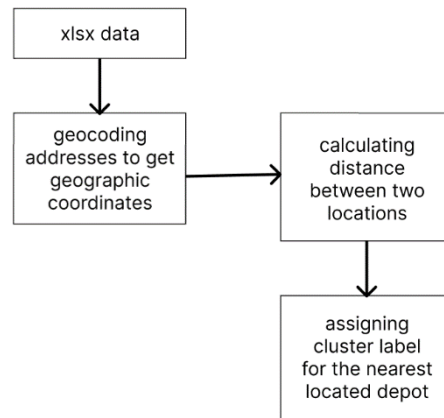


Figure 12 Cluster preparation process

Successful clustering is essential for the optimal route assignment. The clustering approach used in this work is chosen to meet already existing limitations and is based on finding nearest nodes to one of seven cluster centers. These cities will be associated with the cluster and form it:

1. Berlin,
2. Munich,
3. Dresden,
4. Bonn,
5. Duisburg,
6. Hamburg,
7. Stuttgart.

Current depot locations were mapped on the Leaflet map via folium library in the Jupyter Notebook. From the data visualization only a subjective assessment of the distribution and placement of starting points is possible, therefore an additional technique of data cluster analysis is needed.

To evaluate how good depots are located, a K-means clustering from open-source python library scikit-learn was applied. To get predicted centroids for all customer locations we set the default “Lloyd” algorithm, number of clusters to 7, which we want to get and all locations, represented as geographic coordinates. The output cluster centers can be obtained with `kmeans.cluster_centers_` attribute, which returns all generated cluster centroids in an N-dimensional array of geolocations. When mapped together with depots, we can see that current distribution of depots is close to optimal.



Figure 13 Depots' distribution together with K-Means centroids

Therefore, we can expect satisfying clustering results, and expect customer locations to be almost evenly spread over all 7 clusters. Further steps, outlined below, are focused on building routes and optimizing trips, consisting of single routes.

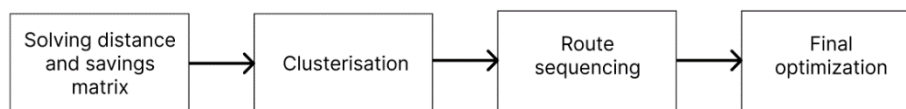


Figure 14 Clustering and optimization phases

When selecting a suitable algorithm, stakeholder requirements should be kept in mind, namely, to reduce travel time as much as possible and to ensure the shortest and best possible route from start to finish. Firstly, it should be possible to set the exact number of clusters to be formed as those already exist and cannot be changed in the near future. Secondly, it is important to have evenly formed clusters, as it will ultimately determine the workload of the employee who is servicing the cluster.

For this reason, the first cluster iteration should exclude nodes located too far away from the centroid. The distance from the depot to the node can be limited to 125 km, which is roughly 1,5 hours ride. All nodes outside the 125 km radius can be added to the most geographically suitable cluster.

To assess the distribution of customer nodes, a manual clustering with above requirements was made. First, the algorithm has to traverse all destinations. All nodes that did not meet the distance criterion, were labeled in the additional column “Longer route”, where “0” means to be in the radius and “1” to be outside. The biggest and most resource-demanding cluster was identified for the “Duisburg” depot. “Stuttgart” cluster has the longest routes to some of its nodes, but it can be explained by the fact that all cities, which were not clearly assigned to any cluster as they were outside the radius boundary, were included in the nearest cluster. This approach is similar to the K-Means method and should be tested for optimality. For this purpose, algorithms based on other clustering rules will be tested and compared as well.

Algorithms for clustering are a part of unsupervised learning, which can discover patterns in the unlabeled data. Given this it is important to keep in mind that there is no absolute “right” or “wrong” answer. The quality of the clustering solution will depend on the specific use case and the

goals of the analysis. Hence, it is common to experiment with different parameters and approaches to find the best solution for a particular problem. Algorithms chosen for analysis are divided into hierarchical, distribution-based, density-based and partitional clustering. Only those were considered, that are suitable to process spatial, in our case geographical, data.

Algorithm	k-means	OPTICS	GMM	Agglomerative clustering
Application objective	Forming clusters with the nearest mean	Detecting meaningful clusters in data of varying density	Clustering and density estimation with data as a mixture of several Gaussian distributions	Each data point – a cluster on its own - gets merged into bigger cluster groups
Clustering type	Partitional	Density-based	Density-based	Hierarchical
Centroids initialisation	Random	Random	Random	Can be set manually
Advantages	Linear time complexity; Number of centroids can be initialized with k-means++ setting	Identifies noise data while clustering	Suitable for modelling complex; Multi-modal distributions quality remains good with time, memory constraints	Good cluster quality after a single scan; Improves the quality with a few additional scans.
Disadvantages	Not suitable for very big datasets; Dependent on initialization of centroid; Impossible to see cluster outliers	Number of clusters depends on radius	Limitations such as sensitivity to initialization; difficulty in estimating the optimal number of components, and difficulty in assigning a data point to a specific cluster	Not suitable for very big datasets, causing $O(n^2 \log(n))$ complexity

Table 8 Comparison of clustering algorithms

Although there is a predefined number $k=7$ of clusters, it is still useful to find the optimal number of clusters, which is achievable by using one of the model selection methods such as BIC or AIC, because the optimal number of clusters is not always clear from the data.

BIC and AIC are model selection criteria used to choose the best model among a set of models with different complexities. These criteria are used in many different machine learning and statistical models, including Gaussian Mixture Models for clustering.

BIC is a measure of the quality of a model that balances the likelihood of the data given the model with the complexity of the model. BIC penalizes models with a large number of parameters, making it a viable choice for models with a large number of candidate models, such as clustering models.

AIC is another measure of the quality of a model that balances the likelihood of the data given the model with the complexity of the model. AIC also penalizes models with a large number of parameters, but is less stringent than BIC, making it a good choice for models with a moderate number of candidate models.

In practice, when choosing the best model among a set of models, BIC or AIC values of the models can be compared to choose the one with the lowest value. The model with the lowest BIC or AIC value is considered to be the best model, as it provides the best balance between the accuracy of the model and its complexity.

The relationship between the number of clusters and the error of the clustering solution can be found by evaluating the performance of the clustering algorithm for different numbers of clusters.

A general outline includes the following steps:

1. Choosing a clustering algorithm: algorithms that can be used are for example K-Means, Hierarchical Clustering, or DBSCAN.
2. Choosing an evaluation metric: the quality of the clustering solution will be evaluated by the metric. Common metrics include the silhouette score, the Calinski-Harabasz index, and the Davies-Bouldin Index.
3. Choosing a range of values for the number of clusters: the initial number of clusters can be small, e.g., 2 or 3, and then increased incrementally.
4. Running the clustering algorithm and clustering quality evaluation by using the chosen evaluation metric.
5. Plotting the results: visualization of results shows the relationship between the number of clusters and the error of the clustering solution.
6. Interpreting the results: interpreting the relationship between the number of clusters and the error of the clustering solution. Suitable methods are the Elbow method or other heuristics to determine the optimal number of clusters.

Below we take a closer look at the definition and application of chosen clustering algorithms and decide on the most fitting approach to solve the problem.

To inspect the quality of each clustering algorithm, we will visualize it on the plot in the Jupyter Notebook using the matplotlib library. Visualization will help examining whether the clusters appear to be well separated and internally homogeneous. Figures visualizing the distribution of the dataset will have following characteristics: data points will be assigned to the cluster and have a distinguishing color, a depot will be marked with a red star for a clear difference from nodes. Axis labels correspond to the latitude and longitude as on the map. Labels for Y axis will represent longitude of the customer or depot geolocation and are displayed reversed on the plot. Labels for X axis stand for latitude of the customer or depot geolocation.

2.7.1. OPTICS

This method falls into the category of density-based algorithms and is similar to the DBSCAN algorithm. Compared to it, it has an advantage of identifying clusters for data with differing density. Although this method wasn't developed for clustering, it shows the structure of the dataset. Its name stands for Ordering points to identify the clustering structure. The algorithm was initially tested with calculated cluster centers.

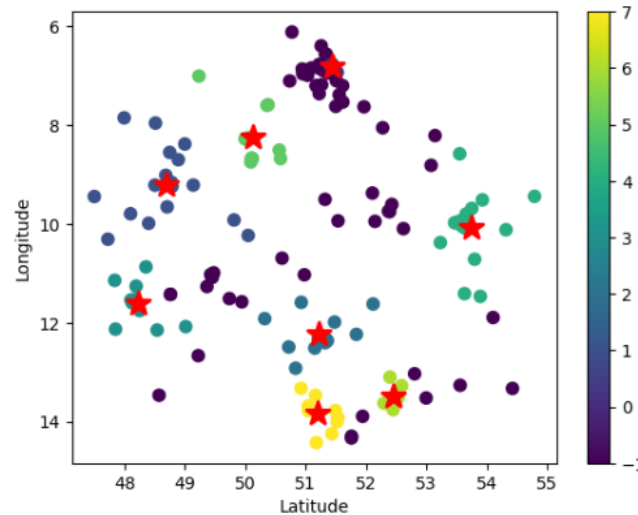


Figure 15 OPTICS clustering result

OPTICS and DBSCAN can be very efficient for clustering and show qualitatively good clustering results, for example in the studies mentioned in the table 7. In our case, clusters are spread unevenly. It is clear that data in the cluster, marked with dark violet color, for the most part was not assigned to any specific cluster and that is the reason for its chaotic distribution. Only those data points with a clear distance similarity were merged into clusters. Due to its density-based clustering logic, this method extracts clusters of arbitrary shape and size by setting a threshold on the reachability distance, which makes it unsuitable to use for our specific problem.

2.7.2. K-Means

This method belongs to the partitional clustering group and has a goal of partitioning the data into clusters by its proximity to the centroids (intra-class similarity). Classical K-Means belongs to the class of unsupervised algorithms and is aimed to find similarities among data and form it into k clusters. K-means can produce different clustering results for the same dataset because it is a stochastic algorithm, meaning that the results can vary each time the algorithm is run, even on the same data. The final clustering result depends on the initial centroid locations, which are randomly assigned in K-Means. If the initial centroids are positioned in such a way that they create uneven clusters, the final clustering result will also be uneven.

To mitigate this issue, a common technique is to run K-Means multiple times with different initial centroids and choose the best solution based on a criterion such as the sum of squared distances between the data points and their nearest centroid. This is known as the "K-Means++" initialization method. This option will be used to set centroids to already existing depots.

Eventually it can be useful to predict best positioning of centroids if the company scales. One method to predict the optimal k number is the Elbow curve method, which is based on the sum of squared distance between data points and clusters assigned to them. It is a heuristic for determining the optimal number of clusters in a clustering algorithm. It is based on the idea that the optimal number of clusters is the one where adding more clusters does not significantly decrease the error or "inertia" of the clustering solution. In the Elbow method, a plot is created that shows the relationship between the number of clusters and the error or inertia of the clustering solution. The number of clusters is increased from 1 to some maximum value, and the corresponding error or inertia is calculated for each number of clusters. The plot of the error or inertia versus the number of clusters is then examined to find the "Elbow point", which is the point at which the error or inertia begins to level off and adding more clusters has a diminishing effect on the error.

The Elbow method is a quick and easy way to get an estimate of the optimal number of clusters, but it is not guaranteed to give the best solution for every dataset. It may not always be clear where the "Elbow point" is, and different datasets may have different shapes to their error versus number of clusters plot, making it difficult to determine the optimal number of clusters.

Based on the distribution of 154 nodes, the optimal number of clusters would be $k=3$, as we can see in the Figure 16. This number cannot be accepted as being too small as the distance between nodes in each cluster would exceed our distance upper boundary and therefore maximum travel duration. A gray line is added to show the elbow the function decreasing at point 3. Greater number of k will not cause any significant change, which is represented by a flattened curve closer to the top. Our initial setting of $k=7$ must be met.

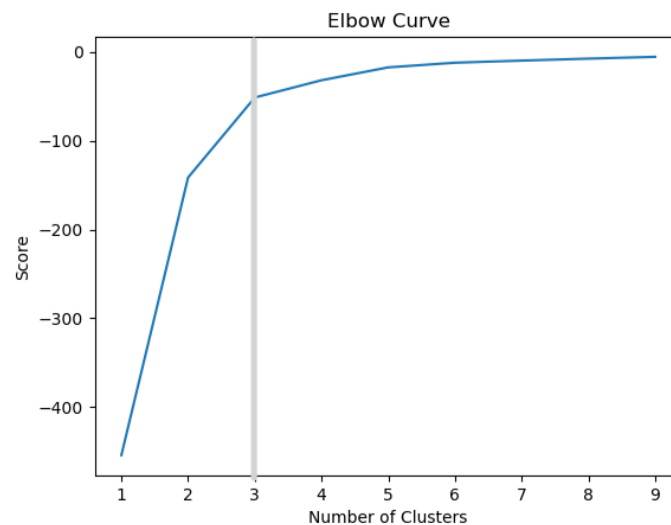


Figure 16 Elbow method

Another method often used to find optimal number of clusters based on the dataset: the DBI method. It is a commonly used evaluation metric in cluster analysis to measure the similarity between the clusters in a clustering solution. The DBI is a scalar value that ranges from 0 to infinity, with lower values indicating more desirable cluster solutions. The DBI is calculated as the average similarity between each cluster and its most similar cluster, where similarity is defined as the maximum similarity between any 2 points in the 2 clusters. The similarity between 2 clusters is given by the formula:

$$\text{sim}(C_i, C_j) = (d(C_i, C_{\text{bar}}) + d(C_j, C_{\text{bar}})) / d(C_i, C_j) \quad (2.1)$$

where C_i and C_j are 2 clusters, C_{bar} is the centroid of the cluster, and d is the distance metric used to calculate the distance between the centroids of the 2 clusters.

In practice, the DBI is used to compare different clustering solutions and select the one with the lowest DBI score. The DBI is sensitive to the number of clusters and the distribution of the data, so it should be used in combination with other evaluation metrics and with a thorough understanding of the problem and the data being analyzed.

To identify the optimal number of clusters for the given dataset a range of $[2,7]$ was set as an input. The output is a DBI value for each k number of clusters.

Number of clusters	DBI value
2	0.9356217481604577
3	0.7602312148279567
4	0.6076015907747261
5	0.6205491058955721
6	0.6588628266310895
7	0.6977902909186143

Table 9 Comparison of DBI values for k number of clusters

According to the DBI method, the most suitable number for clusters is k=4, which is not satisfying our business requirements.

Nevertheless, we can set our predefined custom centers by using the K-Means++ option. This algorithm is best suited to generate a small number of clusters. Compared to K-Means, centroids are not chosen randomly, so there will be no “empty” cluster or 2 multiple centroids in one and the same cluster, which significantly decreases the quality of clustering. For the same reason the time it takes to form a cluster is drastically reduced. Each cluster is represented by its center or so-called centroid which equals the arithmetic average value of all data points added to the cluster. So, the centroid’s value might not be a data point itself. Data points will be iteratively assigned to its nearest cluster. The following distribution is generated by randomly initialized centroids, which were mapped earlier.

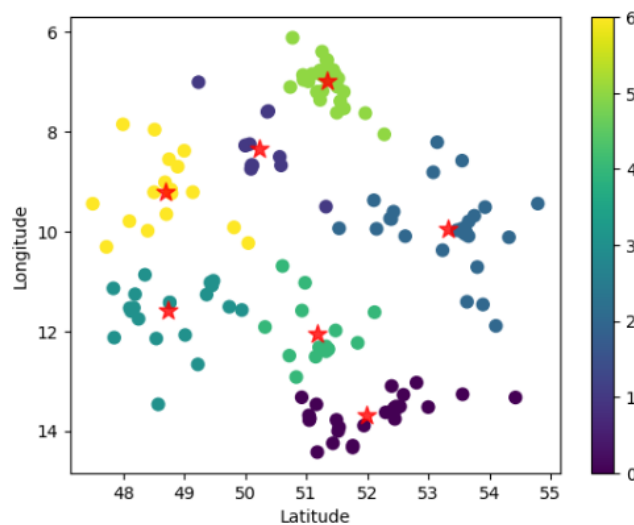


Figure 17 K-means++ clustering result

Following the initialization procedure described above and considering a set of requirements we have, we set our depots as centroids. This way centroids will be in different, separate clusters, each of them containing data points. Clustering has run successfully, outlining 7 centroids and its clusters. K-Means optional parameters include an additional method “minit” for centroids initialization. So, the K-Means model can be initialized with our own centroids numpy array with geographic coordinates. Overall, 6 tests were run and resulted in following clusters:

Test ID \ Cluster	1	2	3	4	5	6
Berlin	22	24	25	21	15	22
Bonn	32	23	30	26	12	32
Duisburg	25	34	16	25	33	26

Test ID \ Cluster	1	2	3	4	5	6
Stuttgart	21	23	23	32	30	25
Hamburg	26	30	33	22	16	13
Munich	15	15	15	15	23	15
Dresden	13	5	12	13	25	21

Table 10 K-Means++ clustering results

Overall, distributions are evenly spread and can be used to form clusters that satisfy problem requirements. Next tests were run with depots that already exist and should be considered for better routing construction within one cluster.

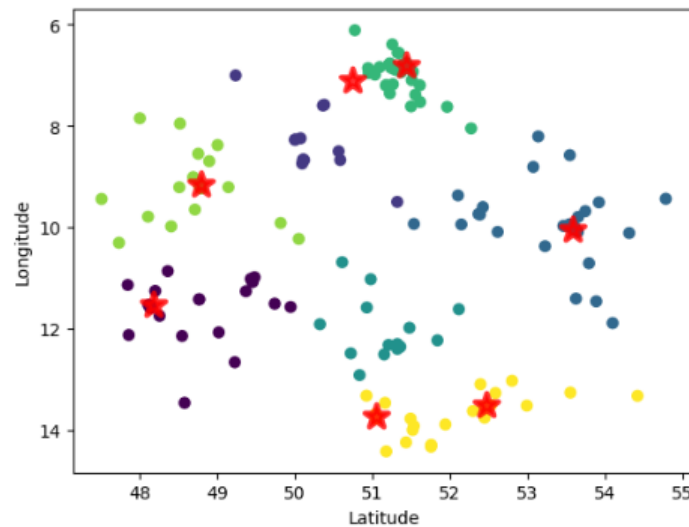


Table 11 K-Means clustering with custom centroids

Running the algorithm showed that clustering remains stable and deliver the same result under same conditions. Below are our initial depots and nodes assigned to them.

Depot	Cluster size
Munich	22
Bonn	13
Hamburg	26
Dresden	15
Duisburg	32
Stuttgart	21
Berlin	25

Table 12 K-Means++ with depots

Important to notice that this algorithm is characterized by the formation of spherical shaped data groups, which might negatively impact clustering outcomes. If clusters are expected to have more complicated geometrical shapes, they will more likely be unsatisfying for further processing. In such a situation Agglomerative Clustering might produce more desirable clusters.

2.7.3. Gaussian Mixture Model

The GMM is a probabilistic model that assumes that the data is generated from a mixture of several Gaussian distributions. Each Gaussian distribution is characterized by its mean, covariance matrix,

and weight. The weight represents the proportion of the data that is generated by each Gaussian distribution.

GMM is a soft clustering method, meaning that each data point is assigned a probability of belonging to each cluster. This contrasts with hard clustering methods, such as K-Means, where each data point is assigned to exactly one cluster.

The GMM can be used for a variety of tasks, including density estimation, anomaly detection, and clustering. To fit a GMM to a dataset, the algorithm iteratively estimates the parameters of the Gaussian distributions (means, covariances, and weights) that best fit the data. This is typically done using the Expectation-Maximization (EM) algorithm.

In terms of clustering, the GMM can be used to find clusters of similar data points by assuming that the data points within each cluster are generated from a single Gaussian distribution. The number of clusters can be specified or estimated from the data.

One advantage of the GMM is its ability to model more complex distributions than other clustering methods, such as K-Means. For example, GMM can model multi-modal distributions, where multiple clusters have different shapes and sizes. Another advantage is that GMM can assign non-uniform weights to each cluster, allowing it to capture the different densities of data points within each cluster.

On the other hand, again compared to K-means, this method can produce different clustering results with the same dataset. If different initial parameters are used, the model may converge to different solutions, resulting in different clustering results.

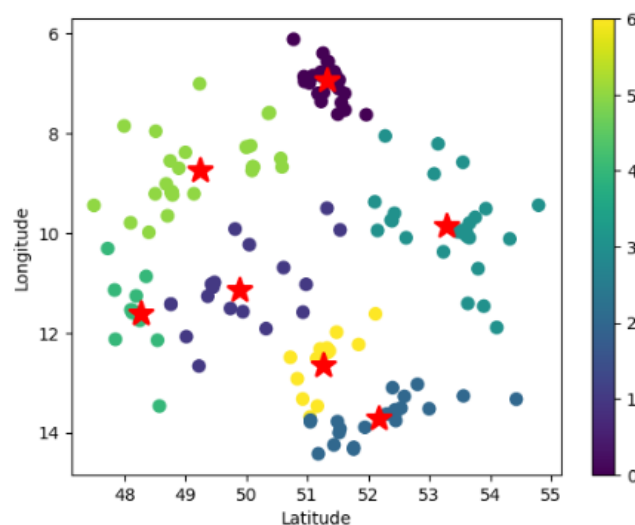


Figure 18 GMM clustering result

The outcome has shown good clustering results, clearly dividing data points into nearest clusters. Nevertheless, resulted centroids are not aligned with preset centroids that already exist and those cannot be changed, so this algorithm is not suitable for the current solution.

GMM provides an option to set own predefined cluster centers. If custom centroids are specified as the initial mean vectors, GMM uses these centroids as the starting point for the estimation of the Gaussian parameters. In this case, the clustering results should be consistent each time the algorithm is run, as the starting point for the estimation of the Gaussian parameters is fixed.

However, it is important to note that the clustering results may still be affected by the choice of the custom centroids. If the custom centroids are not representative of the underlying data structure, the GMM algorithm may produce suboptimal results. Additionally, the choice of the number of Gaussian distributions in the mixture model can also impact the clustering results. Several cluster variations were obtained and evaluated. Below distribution can be considered as a good option.

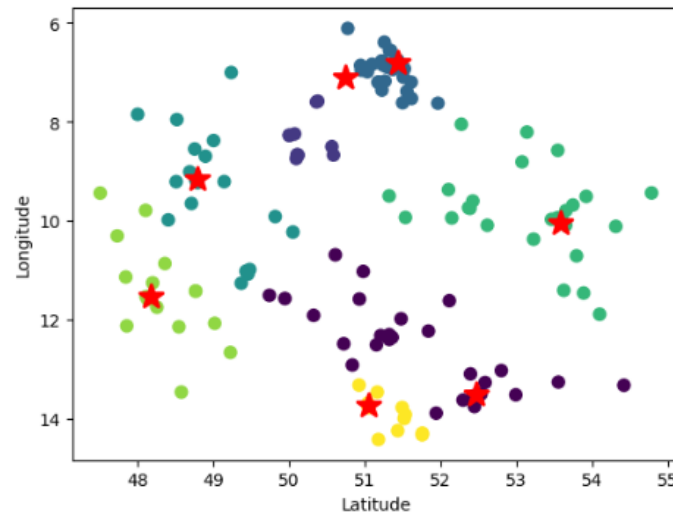


Figure 19 GMM clustering with pre-set centroids

Apart from setting own depots this method can be used to make a prediction for the most optimal location of a new centroid, in case number of depots will grow.

2.7.4. Agglomerative Clustering

The Agglomerative Clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It uses a "bottom-up" approach with each observation being its own cluster. With every merge of a cluster pair the algorithm is building a cluster hierarchy until k number of clusters is obtained. The algorithm is provided with the `sklearn.cluster.AgglomerativeClustering` method and must be configured to meet clustering expectations. The k number of clusters can be set as for `K-Means++` or left as `None`, which sets the $k=2$. The default linkage metric is set to `Euclidean`, which is used for the current implementation. The linkage criterion should be chosen carefully as it can make a major impact on cluster results. The choice of linkage sets the distance, which will be applied on all given observations and eventually determine the shape of each cluster. It will merge those data points that minimize this criterion. Below is the description of each linkage option:

- “average” - uses the average of the distances of each observation of the 2 sets;
- “complete” - uses the minimum distance between all observations in a pair comparison. Produces more “spherical-like” clusters, compared to “single” option;
- “ward” - merges 2 data sets with the minimum variance;
- “single” - is the simplest merging criterion, it uses the minimum distance between all observations in a pair comparison.

Predefined depots were added to the nodes' set to calculate means as well, so the result dataset entails 166 nodes. Below clusters are built for every linkage type with cluster number $k=7$, Euclidean metric and `connectivity=False`, we get different results that can be easily visually evaluated. The “ward” linkage seems to be the most appropriate for our clustering intentions. The clustering with this method is performed in less than a second.

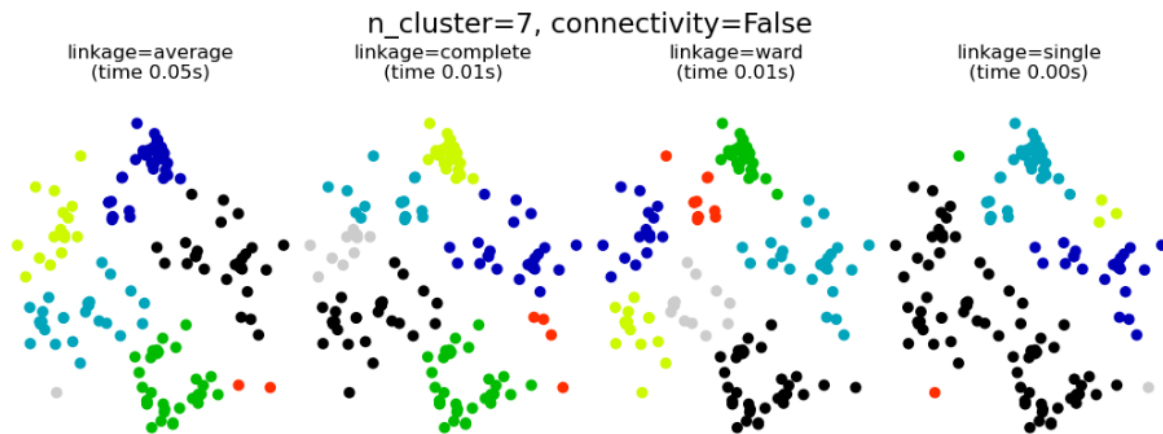


Figure 20 Agglomerative clustering results

With this algorithm we can evaluate how good depots are positioned. Below figure illustrates centroids that are calculated as a mean of a cluster and is marked with a cyan star. Really existing depots are marked with an orange star.

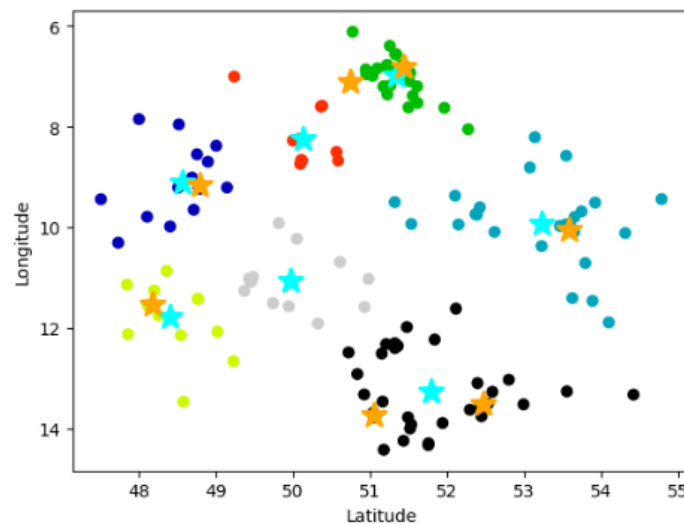


Figure 21 Agglomerative clustering with pre-set centroids

From the illustration above it is obvious that actual depots are not optimally placed for the clusters formed with this particular algorithm, but the nodes' distribution is reasonable and cluster borders are clear to see. Calculated centroids and nodes were assigned to its clusters.

Centroid's optimal location	Number of nodes
Berching	28
Grassau	37
Neritz	16
Heidhausen	35
Hagenburg	15
Niedernhausen	13
Tübingen	22

Table 13 Agglomerative clustering results

2.7.5. Clustering Choice

For more detailed and precise clustering analysis all results of fitting algorithms were represented in the table below.

Cluster ID	City	K-Means++	GMM
1	Munich	22	18
2	Bonn	13	12
3	Hamburg	26	28
4	Dresden	15	36
5	Duisburg	32	31
6	Stuttgart	21	15
7	Berlin	25	14

Table 14 Overview of the distribution for each depot

OPTICS algorithm is applicable to identify clusters of arbitrary shape and size, especially when dealing with complex and noisy data sets. It is based on the density of points, not on their proximity to cluster centers or means. Therefore, it results in clusters that have more diffuse boundaries since the algorithm's grouping of points depends on their relative density to the surrounding points. This approach makes OPTICS effective for clustering data with complex, non-spherical shapes, and varying densities, but it also means that resulting clusters may not have well-defined borders. In our case, clusters built with this algorithm, cannot satisfy our needs as they have too indistinct borders.

We can expect that cluster centers formed with Agglomerative Clustering will not match our needs, because they were originally calculated based on the average of all the data in the cluster. This method can be used for predicting the location of a next depot, if the number of nodes will grow.

GMM and K-Means are both appropriate to cluster geodata with more than 150 locations. From the visualization, K-Means produces better clusters as it assumes clusters are spherical. This approach will be used in the first phase of data clustering.

After selecting the most appropriate clustering method the next phase can be started, namely the route sequencing.

2.8. Route Construction

To solve the symmetric VRP, we have to define a strategy of building routes for each cluster. Solution finding is often split into 2 stages: route construction and route improvement. First stage is designed in a way where one or more rule-compliant routes are iteratively compiled and saved. During the second stage these routes are improved with the chosen local optimization function, the most optimal will be selected. Route construction itself is a process, which can contain various

steps and approaches. If any additional constraints apply, it can be solved iteratively with any modern heuristic. Results in the work of Uchoa et al. point to the fact that classical or so called “exact” algorithms, compared to heuristics or metaheuristics, are not efficient to solve a VRP problem with more than 200 nodes [34]. Therefore, new heuristic approaches were developed to solve large-scaled problems. Some of the examples are Simulated Annealing (1983), Tabu Search (1986), iterated local search. Another way to solve a VRP is to reduce its complexity to a TSP. Both problems, as previously mentioned, are NP-hard, but VRP often contains additional constraints and can be particularly large (with more than 1000 nodes), which can only be solved, if optimal, with multi-phased heuristic or metaheuristic approaches. Taking that into account, the following procedure is proposed: find a single optimal tour, including all cluster nodes and split it into single routes. As mentioned in the work of Beasley - each capacitated VRP can be recycled to TSP [35]. Each cluster VRP can be reduced to TSP with the splitting logic: giant tour will be separated into multiple routes. This method, opposite to the cluster-first route-second, is described in Beasley’s research of 1983, where he outlines the advantages of splitting the overall tour with an optimal procedure called “Split”. An optimal TSP tour can be found with any chosen heuristic as final split paths are subject to the splitting procedure. Work of [36] proves a CVRP tour in the best case can be shortened by half with the splitting procedure. Overall, it is typically used in combination with other methods, such as meta-heuristics or exact algorithms, to find good solutions to these problems.

2.9. Local optimization

The simplest example of a tour construction can be implemented with a NN algorithm, which belongs to greedy algorithms. The tour starts at the depot and, as the name implies, the sequence of nodes is built based on their distance from each other. The tour is constructed when all the nodes are included, returning to the start mode. The implementation of this approach is simple but can result in very long routes at the end, which doesn’t lead to an optimal solution. It can be however improved afterwards. Weiqi Lie states that the running-time factor can be sufficiently improved by the k-interchange heuristic [37]. His solution includes a 2-opt search optimization technique, which is highly effective in local search. Local optimization can be applied to both symmetric and asymmetric neighborhood structure. This heuristic is a great choice for middle to big size problems with more than 100 nodes. It iteratively improves a given solution by disconnecting 2 edges, swapping their positions, and then reconnecting them as long as the resulting route is shorter than the original one. The initial solution can be also a random tour, which will be then optimized by a series of 2-opt moves until it reaches a local optimum. It is guaranteed to find a local optimum, but it may not necessarily find the global optimum, which is the shortest possible route. To improve the chances of finding the global optimum, the 2-opt algorithm can be combined with other optimization techniques. Local optimization can consider not only 2-opt moves, but also 3-opt and k-opt moves. Since real-world scenarios allow for the possibility of sudden changes, there may be a need to improve a section of the route. In the example below a route optimization is achieved by swapping 2 neighboring nodes. It can be especially useful when its edges are crossed.

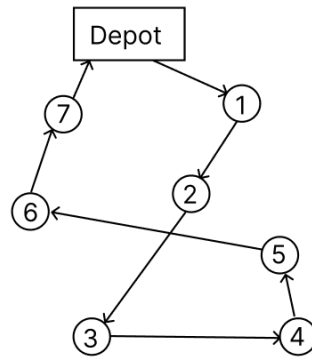


Figure 22 Initial unoptimized route

After swapping edges of pairs of nodes (2, 3) and (5, 6), we get a local optimization resulting in a shorter route.

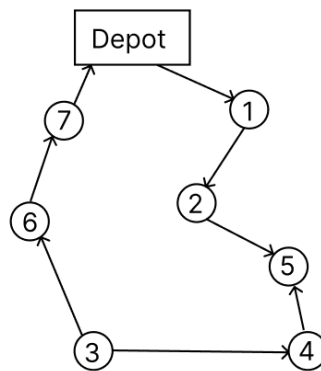


Figure 23 Route after 2-opt swap

The program implementation follows below steps:

1. Call the Solver function on input I and obtain an initial route sequence $S = S_0$.
2. Repeat until S is the most optimal solution found:
 - a. Call subroutine B on inputs I and S .
 - b. If it returns a better solution S' , set $S = S'$.
3. Return S and set it as a final solution.

3-opt is a more advanced version of the 2-opt optimization technique and is used to further improve a solution by swapping three edges in a route. The basic idea behind 3-opt is to find three edges in the route that cross each other and replace them with three new edges that result in a shorter total route length. The 3-opt move can be applied when the current solution to the route optimization problem has a suboptimal structure, such as a long detour or a suboptimal ordering of the nodes. By making small changes to the solution using 3-opt, it is possible to improve the solution and find a shorter route. Like 2-opt, 3-opt can be effective in finding short routes, but it is not guaranteed to find the optimal solution.

Overall, 2-opt is a good starting point for solving the TSP or VRP and is simple to implement but may get stuck in a local optimum. 3-opt can improve the solution found by 2-opt but is more computationally expensive. Another algorithm that is based on 2-opt and is often applied for local

optimization, was implemented by Brian W. Kernighan and Shen Lin in 1973 and is called a Lin-Kernighan (LK) algorithm. It is considered the best for solving TSP. In general, LK is considered to be the best [38] of the three for finding optimal solutions, but it is also the most computationally expensive. It is a combination of the greedy construction heuristic and the local search heuristic. The algorithm starts with a random solution and iteratively improves it by swapping the position of 2 nodes in the route. The algorithm terminates when no further improvement can be made to the current solution.

To compare some of the simple approaches that can be easily implemented for the solution, a case scenario for 13 nodes was created. This specific number of nodes is set as a benchmark, as it is a size of the minimal cluster, thus the tests should be successful in order to be stable for bigger clusters. Below each scenario is run with an algorithm, comparing total time and distance required to traverse all of the cities in the TSP route. Calculation time is expressed as a function of the size of the input, usually denoted by "n" (i.e., the number of cities in the TSP problem), time complexity refers to the actual amount of time taken by the algorithm to solve the TSP problem for a given set of inputs.

Algorithm	Total traveling time (hours)	Total km	Calculation time	Worst time complexity
Brute-Force	12.6	686.6188	4 hours, 48.73 seconds	$O(n!)$
K-Nearest Neighbours	15.67	879.6279	3734 microseconds	$O(n \log n)$
Lin-Kernighan and k-opt	12.85	688.0483	16778 microseconds	$O(n^2)$

Table 15 Case scenario test

It is clear that the LK algorithm produces an optimal solution, which is indicated by the travel duration and route distance. A Brute-Force and its solution as global optimum takes significantly more time than other algorithms and is therefore not suitable for the task, but rather fits problems with smaller set of nodes. Scenario testing has proven the effectiveness of LK approach. Keld Helsgaun demonstrates effectiveness of his TSP-heuristic implementation, where k-opt is complemented by LKH-2 (Lin-Kernighan algorithm variation) [48]. The computation complexity was significantly minimized to be almost linear $O(n)$, that is proportional to the input size. This hybrid algorithm was chosen to be used to solve the formulated problem.

3. Technologies for the Implementation

As a result, the following optimization framework is expected to be efficient for the formulated problem and is to be realized with the proposed algorithm. It excludes the clustering process and describes only the route construction and optimization phases. Cluster formation can be applied when a customer or depot locations' dataset was changed and requires update. In this case a cluster algorithm based on K-Means method can be run, resulting in .csv files containing locations assigned to each cluster.

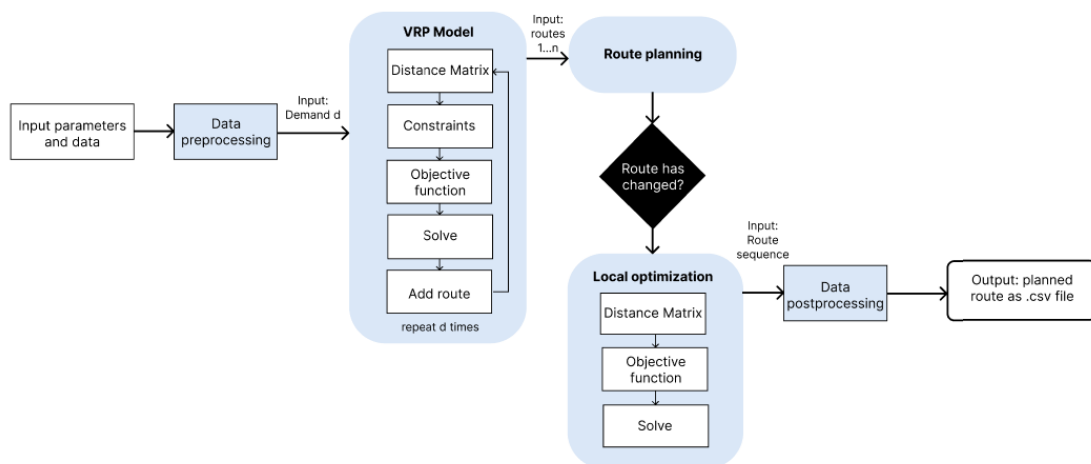


Figure 24 Optimization framework

The implementation is to be designed with a user-friendly interface, so that every employee could use it. It is intended to be used by the operational manager in planning weekly trips. The initial file required for input, is based on the file system and standards of the company. The .csv or .xlsx file contains multiple columns, relevant to the operations team, like the address of the customer, number of service units booked etc. This file is constantly and manually maintained and updated by the employee. Data processing step happens in the Jupyter Notebook environment to eliminate false or duplicated data. Cleaned data is then used to construct multiple routes in advance for chosen customers in one particular cluster. Ideally it can also dynamically update a route when a change in the route needed. Its construction is based on constraints and objective function: minimizing the overall trip distance. The found solution goes through the postprocessing step in which it is brought to the readable format such as a .csv file or visual schedule representation on the screen for the user.

The choice of technological stack was made in favor of rich open-source and established software. Operations research tools of Google (OR-tools) provide free access to its software libraries and APIs with minimal restrictions to registered users. They are commonly used for constraint optimization, linear optimization, and flow and graph algorithms [39].

Python programming language and Jupyter Notebook make a natural combination for major functionality we would need such as data cleaning, filtering, visualization, and analysis. We also expect the IDE to be ideal for the number of calculations needed while data preparation and experimenting itself.

This technological stack has proven itself to be a rational choice as it was used in the scientific literature devoted to the VRP, for example by Kumar and Munagekar [40].

The interface is implemented with a Flask web application framework, which is a good choice for building web interfaces in Python. Flask is a lightweight and flexible web framework that allows developers to quickly build web applications with minimal boilerplate code. It provides a simple and intuitive way to handle HTTP requests and responses and supports many popular web development features such as URL routing, template rendering, and session management. Flask is also highly customizable, which means that developers can easily add their own functionality and third-party libraries to their Flask applications. Additionally, Flask has a large and supportive community of developers who provide many useful extensions and plugins to enhance the framework's capabilities. Overall, Flask is a powerful and versatile web framework that can be used and is enough for a web interface of our implementation. Implemented web UI can be seen in appendix D.

Following input is required to ensure data processing in the Jupyter lab development environment and further export in a document file:

input needed from the user:

- customer and depot locations in .csv files;

The .scv file for drivers locations (file name used in the implementation: “coaches.csv”) was converted to a DataFrame “coaches” for further manipulation and has a following structure:

Column name	Coach	City	Latitude	Longitude	Address
Data type	string	string	float	float	string

Table 16 Data structure for depots DataFrame

- nodes chosen for route construction;
- date, on which the route should start (“dd-mm-yyyy” format).

Data retrieved from imported files and input information from the user:

- distance and time matrices, built for all chosen cluster nodes;
- customer nodes’ geolocation;

A sheet from .xlsx file for customer locations (file name used in the implementation: “e2023.xlsx”) was extracted in order to convert it to a DataFrame “df_clean” for further manipulation. Final DataFrame has a following structure, where “Kunde” represents a customer company name, “Standort” the name of the customer geolocation, “PLZ” is a postal code:

Column name	Kunde	Standort	PLZ	Check-ups	latitude	longitude	address
Data type	string	string	integer	float	float	float	string

Table 17 Data structure for depots DataFrame

- days labeled with data and status of either working or weekend day.

Minimal expected output as .csv file including:

- best found route path, consisting of node sequence;

- total route travel time;
- total route length in km.

4. Solution Approach

In this chapter the route sequencing step will be tested in cases of different scale and complexity. Solution results will be first recorded and compared before applying the previously defined approach of k-opt in combination with Lin-Kernighan algorithm. Its performance and adaptability will be first tested on a bigger number of nodes - at the moment the calculation time will take too long, for example more than 20 seconds. It will be applied on the case with continuously growing complexity, thus increasing the number of nodes to visit and daily time limitations. The heuristic approach mentioned above is an implementation of research fellow at Monash university in Melbourne, Arthur Mahéo [41]. As mentioned by the author, his algorithm implementation is based on the report of K. Helsgaun.

4.1. Route with no Constraints

To break down the complexity, we will start with a real use case, which can be solved as a TSP. The solution should provide the most optimal route in terms of distance. At this stage we don't have any capacity or time constraint, for the input we need to know the nodes to visit and a depot.

Use Case 1: visiting four nodes labeled with the cluster "Hamburg".

Description: four full working days, visiting four clients total.

Requirements:

- each node should be visited by the driver before going back to the depot;
- each node is visited only once;
- the route is built for one vehicle;
- route should start and end in the depot;
- route should be the most optimal (short).

Below are nodes to be visited and represented by its cities. When calculating distances, more accurate addresses will be used, and distances retrieved with precise geolocation input:

1. Office location in Hamburg;
2. Bad Doberan;
3. Schwerin;
4. Wismar;
5. Lübeck.

Below are the nodes and depot illustrated on the map. In the next steps we will define a strategy to find the optimal route and represent it in a visually appealing way.



Figure 25 Nodes location

As TSP has been widely researched over the last 90 years, there is a variety of classical algorithms and its variations. With a small number of nodes, we can use an implementation which traverses all possible routes, which start and end in the depot. This can be accomplished with a Brute-Force Algorithm, which is suitable for a rather small number of nodes and in this case able to generate a solution within a reasonable period of time. One route consists of multiple trips. For now, there are no constraints for a trip, except that only one trip can be done per day. The solution should be simple to understand and guarantee the most optimal result possible. The algorithm of searching for the best route can be described as follows:

1. Set the first node in the matrix as a depot - starting and ending point.
2. Generate all possible permutations (routes) of nodes excluding the depot, using iter.tools library.
3. Add up all distances of each permutation and save the last found minimum sum.
4. Return the permutation sequence and its sum value.

The distance matrix is calculated with a separate `dist()` function, which takes a pair of geolocations as input and returns the circle distance between them, applying a haversine formula. This formula is more accurate for geolocations, in comparison to a Euclidean distance, which draws a straight line between 2 points.

	Depot	Bad Doberan	Schwerin	Wismar	Lübeck
Depot	0	134.2802	90.4428	99.4743	49.8975
Bad Doberan	134.2802	0	61.4851	36.6110	84.4292
Schwerin	90.4428	61.4851	0	29.3484	49.4468
Wismar	99.4743	36.6110	29.3484	0	50.4468
Lübeck	49.8975	84.4292	49.5786	50.4468	0

Figure 26 Distance matrix

Result returned is a minimum value of 288.8832 km for the route [0, 2, 1, 3, 4, 0]. Nodes will be visited in the following sequence: Hamburg - Schwerin - Bad Doberan - Wismar - Lübeck - Hamburg.

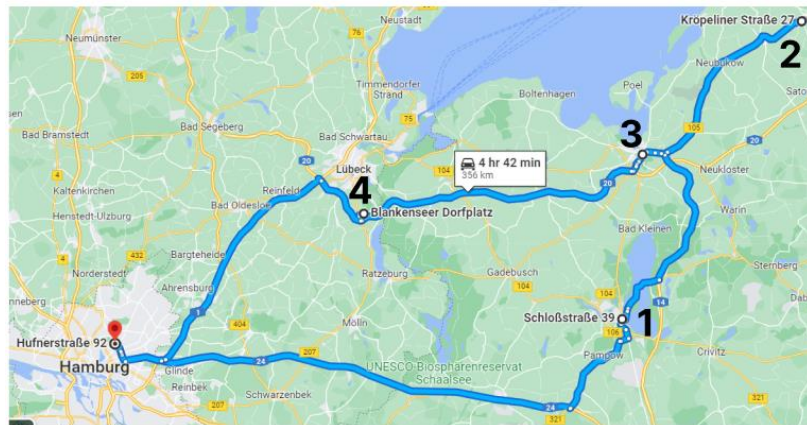


Figure 27 Result of TSP approach

Tests were run thirty times and their times were saved in an array `time_arr`. Average time for building the route for this case resulted in 262.7 microseconds. The mode is however null, so although the algorithm traverses all the routes possible, for a rather small problem it works really fast, satisfying our requirements.

To get more precise results we will use Google OR Tools to build a more accurate distance matrix and prove if the route is truly most optimal. It provides a more comprehensive overview of the distance and time amounts, as an API request is getting live data using geo coordinates of origin and destination addresses. From the API response we then retrieve needed distance information and build a new matrix, which we use as an input for our TSP function.

	Depot	Bad Doberan	Schwerin	Wismar	Lübeck
Depot	0	166.674	108.459	124.829	69.304
Bad Doberan	180.382	0	77.893	42.293	114.786
Schwerin	107.537	76.238	0	45.025	75.141
Wismar	124.718	42.921	49.562	0	59.122
Lübeck	69.491	101.77	75.174	59.925	0

Figure 28 Distance matrix from API response

With the new matrix we get the most reliable data which proves that the suggested route is indeed the most efficient, which results in 355,603 km, which is pretty accurate, compared to the total distance we get on www.google.de/maps.

Time matrix with values in hours shows the time duration for traveling between each of the 2 nodes and it satisfies our requirements as long as routes last less than 2 hours in one direction.

	Depot	Bad Doberan	Schwerin	Wismar	Lübeck
Depot	0	1.93	1.45	1.44	0.87
Bad Doberan	1.92	0	1.11	0.71	1.17
Schwerin	1.41	1.05	0	0.66	1.02
Wismar	1.38	0.73	0.69	0	0.64
Lübeck	0.84	1.17	1.04	0.68	0

Figure 29 Duration matrix from API response

Total time duration of the route according to the google API lasts 4,69 hours. In later steps, when more nodes are added, the algorithm can be optimized in a way it builds the route considering the upper travel duration boundary, in order to drop most uncomfortable routes for drivers.

4.2. Date Assignment

A date assignment described here is not a classical time constraint, which is typical for the VRP with time windows, but is useful to be able to plan and generate a timetable. We want to assign each working day a specific date, excluding public holidays and weekends (Saturday and Sunday). In python we can use an open-source library “holidays” to filter days which are officially holidays, also a python “weekday” function of class date, which returns the day of the week as an integer. For example, zero stands for Sunday and five stands for Saturday. Later it will be possible to build longer routes, for example, which last several weeks.

We have created a Pandas DataFrame - a two-dimensional table-like data structure with labeled axes (rows and columns) to store and manipulate data. Here we define seven columns:

1. "start" - node where a single trip starts.
2. "destination" - node where a single trip ends.
3. "duration" - travel time between 2 nodes in hours.
4. "km" - distance between 2 nodes in km.
5. "day" - date of each day of the travel route.
6. "week" - integer (numbers 1-4) for a week number.
7. "weekend" - binary value (0 or 1) defines if the date is a weekend or a working day.

For input we need a start date, on which the route will start, further days will be generated and added to the DataFrame accordingly. Example for the route starting on 19-12-2022, which is Monday.

	start	destination	duration	km	day	week	weekend
0	Depot	Wismar	1.45	90.4428	2022-12-19	1	0
1	Wismar	Schwerin	1.05	61.4851	2022-12-20	1	0
2	Schwerin	Bad Doberan	0.71	36.611	2022-12-21	1	0
3	Bad Doberan	Lübeck	0.64	50.4468	2022-12-22	1	0
4	Lübeck	Depot	0.84	49.8975	2022-12-23	1	0

Figure 30 Generated use-case plan

For comparison we input the date of the next day, 20-12-2022, as a start day. We can see that labels for the “weekend” column changed accordingly, as days from 24-12-2022 to 26-12-2022 are public holidays and rows were filled with almost no data. A designated column “week” shows the order of weeks, increased by one, once a value in a column “day” is Monday. Work-free days were filled with 0 in the “km” column to show no work trips should take place on holidays.

	start	destination	duration	km	day	week	weekend
0	Depot	Wismar	1.45	90.4428	2022-12-20	1	0
1	Wismar	Schwerin	1.05	61.4851	2022-12-21	1	0
2	Schwerin	Bad Doberan	0.71	36.611	2022-12-22	1	0
3	Bad Doberan	Lübeck	0.64	50.4468	2022-12-23	1	0
4	-	-	-	0	2022-12-24	1	1
5	-	-	-	0	2022-12-25	1	1
6	-	-	-	0	2022-12-26	2	1
7	Lübeck	Depot	0.84	49.8975	2022-12-27	2	0

Figure 31 Plan with opt-in weekends

Now it is important to add the next constraint, which does not allow the driver to stay at the destination over weekends or public holidays. In this situation, the driver should always come back to the depot. On the last working day before the weekend or holiday, the row will be marked with the same date as previous day as the driver is going back to the depot after performing a service to the customer. For that purpose, a new variable `last_node` will be added to save its index. Therefore, an additional rule is needed, which can be described by following code, where `isItHoliday(date)` function checks whether the next day is a public holiday or a weekend and if so, it duplicates the previous date and calculates the distance between the current node and the depot:

```
if isItHoliday(next_day) == True:
    last_date = last_node.getLastWorkingDay() #get last date saved in the dataframe
    new_row = dataframe.addNewRow() #add new row to the dataframe
    new_row["day"] = last_date # save the date in the new row
    new_row["km"].append(getDistanceBetween(last_node, Depot) # save distance
```

Listing 1 Function function for defining weekends and weekdays

After adding these rules, we get a clearer view of the daily trips we have in the route as shown in the Figure 32, including distance and time duration, retrieved with google API. The driver is going back to the depot at the same working day when the last service was provided. With the python module "datetime" we can easily manipulate the date, visualize it and use it for easier planning.

This way we get a more readable and clearer schedule, indicating weekends and holidays as non-working days. On these days no trips are scheduled therefore start and destination locations are marked as depot. Travel distance in km and time duration will be marked in its columns accordingly.

	start	destination	duration	km	day	week	weekend
0	Depot	Schwerin	1.45	90.4428	2022-12-20	1	0
1	Schwerin	Bad Doberan	1.05	61.4851	2022-12-21	1	0
2	Bad Doberan	Wismar	0.71	36.611	2022-12-22	1	0
3	Wismar	Lübeck	0.64	50.4468	2022-12-23	1	0
4	Lübeck	Depot	0.87	49.8975	2022-12-23	1	0
5	Depot	Depot		0	2022-12-24	1	1
6	Depot	Depot		0	2022-12-25	1	1
7	Depot	Depot		0	2022-12-26	2	1
8	Lübeck	Depot	0.84	49.8975	2022-12-27	2	0

Figure 32 Visual plan improvement

In this case, total route traveling distance, calculated with haversine formula, equals 338.780 km. With google API we get (in the given moment) a total of 424.907 km. From now on we will save distances in the "km" column, which were calculated with the google OR tools.

4.3. Route with more Nodes

Next, we simulate constructing routes for more nodes, considering holidays or weekends.

4.3.1. Planning a Weekly Route

We will test the robustness of the current implementation with more nodes, adding six more nodes, which results in 11 nodes and a 11x11 distance matrix. Following dictionary was used for the implementation:

```
city_names = {0:"Depot", 1:"Bad Doberan",2:"Schwerin",3:"Wismar", 4:"Lübeck",
5:"Kobrow",6:"Mühlen Eichsen",7:"Greven",8:"Hevenbruch", 9:"Klützn", 10:"Hohen Sprezn"}
```

Listing 2 Dictionary with city names as input

This data was used for testing purposes only and is generated within the Solver function as input for the optimization part. These indexes, here - keys, will be used to identify each particular node. In the final implementation customer or depot geolocation data will be retrieved and processed dynamically from the .csv file.

Extended TSP algorithm accepted 11 nodes and built the best route [0, 7, 6, 2, 5, 10, 1, 3, 9, 4, 8, 0] with 351.1295 km total traveling distance. This time route calculation took longer - 37.72 seconds on average.

As the computation time rises with the number of nodes, a more efficient solving algorithm should be used. The solution can be found much faster with the k-opt, optimized by the Lin-Kernighan heuristic. It is more stable for big-sized problems and can be edited with additional constraints. The algorithm procedure consists of 2 phases: first, a 2-opt move is made, which is swapping action of 2 edges that optimizes the route. Number of swapped edges can be increased to 3-opt and more until there is no optimization possible. With the Lin-Kernighan heuristic the precise number of moves can be defined. It is known that route length N is optimal if no N -opt improvement move exists. This number of moves will not be covered due to complexity $O(n^n)$.

Lin-Kernighan approach considers smaller moves, that constitute the k-move and includes the lowest possible, while selecting the maximal k. The final solution includes new moves only in case of the optimization, thus saving computation time. For the input it needs a start depot.

This approach was tested with the previously tested set of nodes ($N=11$) and succeeded to build the same route, which is the most optimal, and showing significantly smaller time complexity: it took less than 0,0022 seconds on average.

Below case is tested with another start_date input "2022-12-22". As the route should be interrupted by going back to the depot because of the holidays, the total distance has increased from 338,780 km to 486.0267, that is additional 147,25 km. It makes the built route suboptimal and does not satisfy the requirement of minimizing the total travel cost. The trip distance from Bad Doberan and Depot is the biggest and should be avoided. Thus, an additional refinement step, for example local optimization or integer linear programming technique, is required.

	start	destination	duration	km	day	week	weekend
0	Depot	Schwerin	1.45	90.4428	2022-12-22	1	0
1	Schwerin	Bad Doberan	1.05	61.4851	2022-12-23	1	0
2	Bad Doberan	Depot	1.93	134.2802	2022-12-23	1	0
3	Depot	Depot	0	0	2022-12-24	1	1
4	Depot	Depot	0	0	2022-12-25	1	1
5	Depot	Depot	0	0	2022-12-26	2	1
6	Depot	Wismar	1.44	99.4743	2022-12-27	2	0
7	Wismar	Lübeck	0.64	50.4468	2022-12-28	2	0
8	Lübeck	Depot	0.87	49.8975	2022-12-29	2	0

Figure 33 Plan generated including long weekends or holidays

The most optimal path with the existing k-opt and Lin-Kernighan algorithm can be obtained in case there are no weekends or holidays that can interrupt the planned route. Former can be avoided with a weekly planning, starting on Monday or Tuesday and ending Thursday or Friday accordingly, so that no weekends are included. A proposed relaxation is therefore to start weekly planning on Monday or if it is a holiday, on Tuesday or any following day otherwise. The latter cannot be

avoided unless the route is optimized by another approach. Below we modeled 2 scenarios for a week, the first one starts on holiday and the second starts on a working Monday.

First scenario with the start day as holiday on 26 December 2022.

	start	destination	duration	km	day	week	weekend
0	Depot	Depot	0	0	2022-12-26	1	1
1	Depot	Schwerin	1.45	90.4428	2022-12-27	1	0
2	Schwerin	Bad Doberan	1.05	61.4851	2022-12-28	1	0
3	Bad Doberan	Wismar	0.71	36.611	2022-12-29	1	0
4	Wismar	Lübeck	0.64	50.4468	2022-12-30	1	0
5	Lübeck	Depot	0.87	49.8975	2022-12-30	1	0

Figure 34 Plan generated for the start date as holiday or weekend

Second scenario with the start day as regular day on 2 January 2023.

	start	destination	duration	km	day	week	weekend
0	Depot	Schwerin	1.45	90.4428	2022-12-19	1	0
1	Schwerin	Bad Doberan	1.05	61.4851	2022-12-20	1	0
2	Bad Doberan	Wismar	0.71	36.611	2022-12-21	1	0
3	Wismar	Lübeck	0.64	50.4468	2022-12-22	1	0
4	Lübeck	Depot	0.87	49.8975	2022-12-23	1	0

Figure 35 Plan generated for regular working day as start date

Thus, total traveled distance remains the same and the algorithm is adapted for weekly planning with or without holidays.

4.3.2. Local Paths' Optimization

Next, we want to see if the algorithm can be adapted for longer routes, which involves two or more weeks planning. The main goal is to optimize an algorithm if its initial planning is disrupted by going back to the depot due to the weekly working time overload. This sequence can be visualized as an array of indexes. The new route consists of more nodes – 10 and a depot. A sequence generated for TSP with the Lin-Kernighan algorithm implementation, would take 351.1295km total. Now the simulation will include weekends or holidays, on which the driver has to go back to the depot. Therefore, the route sequence will be divided into 3 parts: {1: [0, 7, 6, 2, 5, 10, 0]}, {2: [0, 1, 3, 9, 4, 0]}, {3: [0, 8, 0]}. Keys 1-3 indicate a calendar week and array numbers indicate geolocations, which they get according to their initial position in the list related to the particular cluster. Zero represents a depot. Below we can see the splitting of the route by weeks.

	start	destination	duration	km	day	week	weekend
0	Depot	Greven	0.81	47.6776	2022-12-19	1	0
1	Greven	Mühlen Eichsen	1.04	45.8849	2022-12-20	1	0
2	Mühlen Eichsen	Schwerin	0.55	16.2228	2022-12-21	1	0
3	Schwerin	Kobrow	0.67	29.4335	2022-12-22	1	0
4	Kobrow	Hohen Sprenz	0.72	37.9518	2022-12-23	1	0
5	Hohen Sprenz	Depot	1.99	146.0775	2022-12-23	1	0
6	Depot	Depot	0	0	2022-12-24	1	1
7	Depot	Depot	0	0	2022-12-25	1	1
8	Depot	Depot	0	0	2022-12-26	2	1
9	Depot	Bad Doberan	1.93	134.2802	2022-12-27	2	0
10	Bad Doberan	Wismar	0.71	36.611	2022-12-28	2	0
11	Wismar	Klützt	0.38	16.1403	2022-12-29	2	0
12	Klützt	Lübeck	0.73	37.8965	2022-12-30	2	0
13	Lübeck	Depot	0.87	49.8975	2022-12-30	2	0
14	Depot	Depot	0	0	2022-12-31	2	1
15	Depot	Depot	0	0	2023-01-01	2	1
16	Depot	Hevenbruch	0.8	33.967	2023-01-02	3	0
17	Hevenbruch	Depot	0.8	33.967	2023-01-03	3	0

Figure 36 Plan generated for a longer time period

One approach that can be applied here can be seen as a local search and optimization, in which each single path, consisting of more than 3 nodes, will be used as an input for the local iterative optimization. The function we used for the initial route can be used for its parts as well to see if a shorter path can be found. It can be simply achieved by the enumeration of nodes' permutations or Brute-Force algorithm, which always finds the best permutation among all generated. Its code implementation was taken from [42] and is listed in the appendix A. It will not affect the calculation time much, as the search space remains low. Function accepts a distance matrix and number of nodes to build the route as an input. The difference after applying the optimization for this route is shown in Figure 37, route visualization is shown in Figure 38.

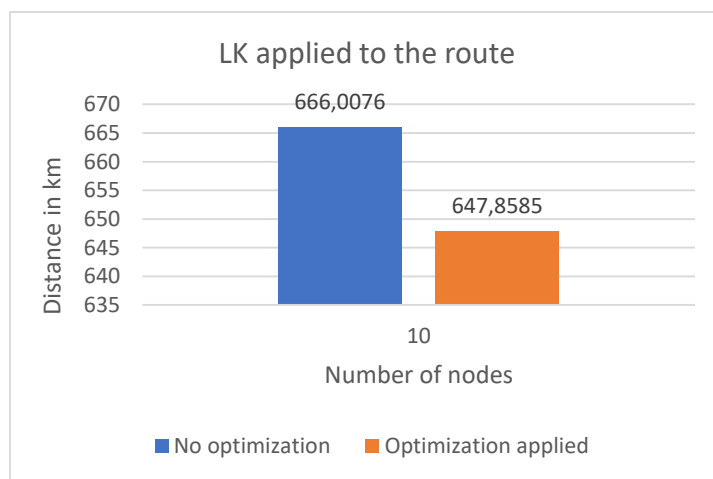


Figure 37 LK applied to the route with 10 nodes

The improvement of the path after optimization resulted in 2.725 % or 18.15 km. Paths are visualized below: first path is coloured red, second – with blue colour, third – black colour.

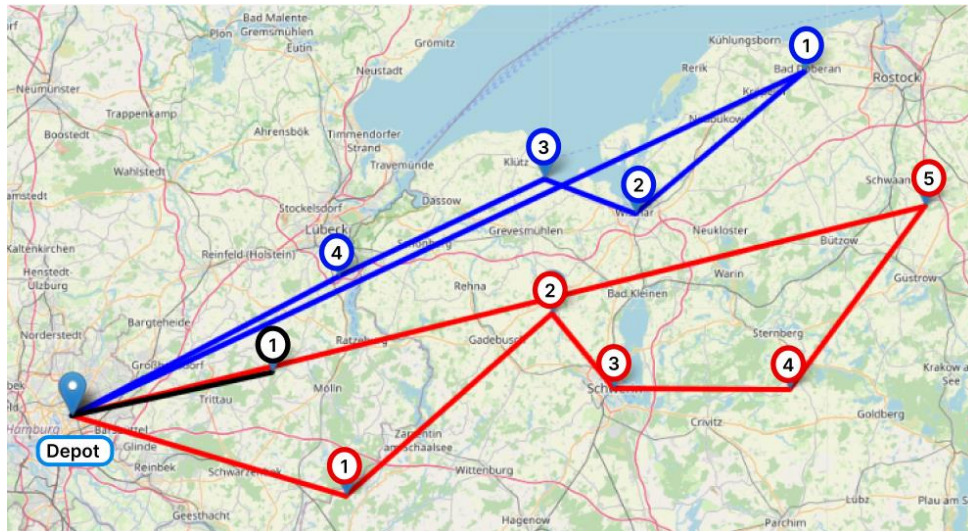


Figure 38 With Lin-Kernighan generated route

To test a random approach each time paths' initialization was run 1000 times and the shortest route permutation among them was found. For random paths' generation there are total of 10! nodes' permutations possible which is 3,628,800 variations. All routes' total distances were calculated and sorted. The shortest route has resulted in the bigger overall route distance compared to the approach of combining Lin-Kernighan and Brute-force algorithms proposed in this thesis. Shortest randomly generated route is marked green in the table below.

Test ID	VRP paths	Total km
1	[0, 4, 10, 1, 2, 7, 0], [0, 3, 5, 9, 6, 0], [0, 8, 0]	699.6973
2	[0, 2, 10, 1, 9, 6, 0], [0, 8, 3, 5, 7, 0], [0, 4, 0]	693.8483
3	[0, 8, 2, 6, 3, 7, 0], [0, 5, 1, 10, 9, 0], [0, 4, 0]	697.5169
4	[0, 2, 3, 1, 10, 9, 0], [0, 8, 6, 5, 4, 0], [0, 7, 0]	683.5454
5	[0, 7, 2, 10, 5, 4, 0], [0, 9, 6, 3, 1, 0], [0, 8, 0]	692.1587
6	[0, 10, 1, 5, 3, 6, 0], [0, 4, 9, 2, 7, 0], [0, 8, 0]	659.5953
7	[0, 7, 6, 1, 10, 9, 0], [0, 4, 5, 2, 3, 0], [0, 8, 0]	686.263
8	[0, 2, 10, 5, 6, 7, 0], [0, 8, 3, 1, 9, 0], [0, 4, 0]	694.448
9	[0, 4, 3, 5, 10, 9, 0], [0, 7, 6, 1, 2, 0], [0, 8, 0]	699.8399
10	[0, 6, 1, 10, 5, 8, 0], [0, 4, 2, 3, 9, 0], [0, 7, 0]	656.1643

Table 18 Best path acquired with random generation

Paths of the shortest route found are visualized below: first path is coloured red, second – with blue colour, third – black colour.

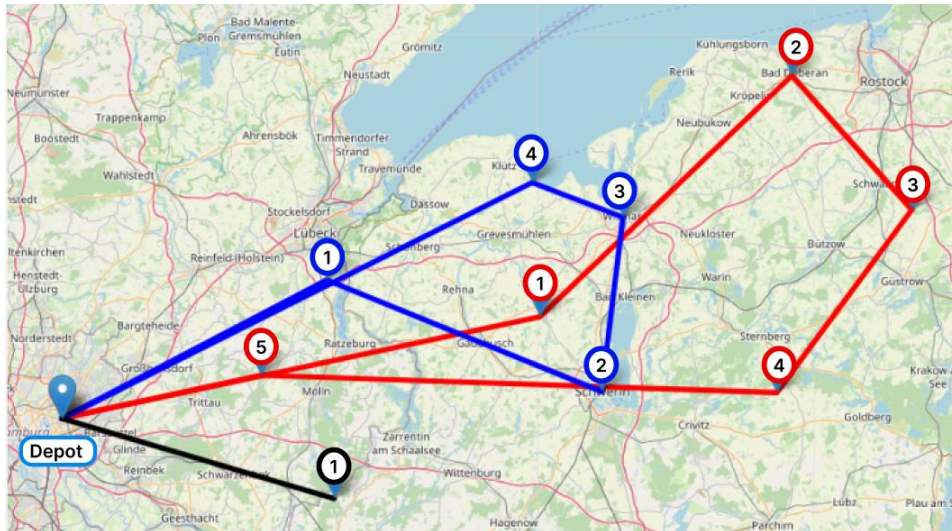


Figure 39 Best randomly generated route

A NN algorithm was implemented for testing its quality and comparison to other approaches. Function `get_neighbors()` accepts a distance matrix of all nodes and returns a sequence of visited nodes and a list of distances which are then summed up to get the total route length. Its implementation can be found in the Appendix B.

Route generated with the NN algorithm was constructed once as a starting point is set to depot and remains the same. Hamiltonian cycle or completed graph built with this algorithm resulted in the route with following paths' indexes: [0, 8, 4, 6, 2, 3, 9, 1, 10, 5, 7, 0]. The total route distance is 389.6222 km. After applying splitting procedure considering as previously weekends and holidays, we get paths with lengths of [234.5517, 322.0256, 95.3552] and more than the doubled overall route distance of 651.9325 km as driver has to go back to depot. Routes obtained after Split procedure are visualized below: first path is coloured red, second – with blue colour, third – black colour.

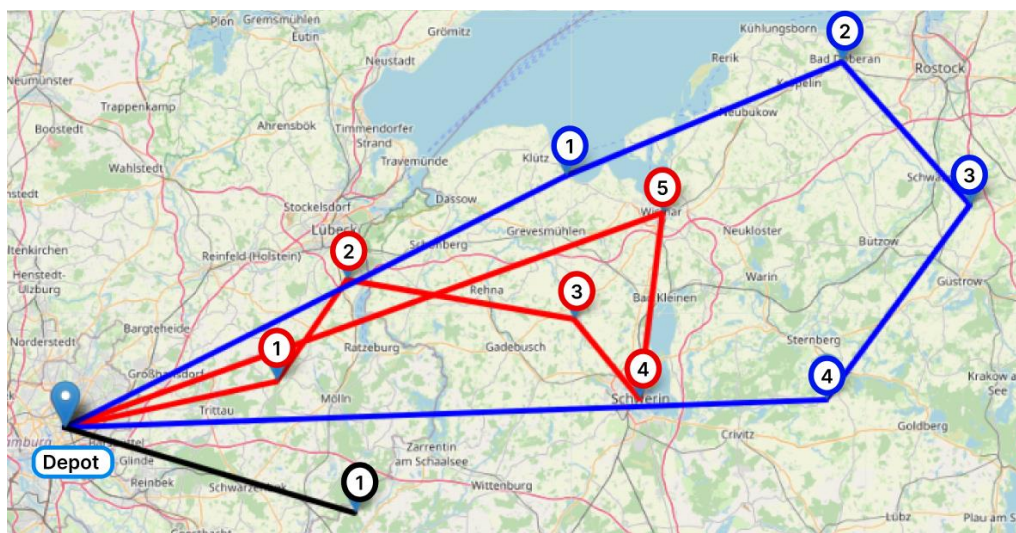


Figure 40 With NN generated route

In order that splitting has the least negative effect on the overall route length, specific constraints and requirements of the problem instance should be evaluated in the context of the overall optimization objective. It can be used in perspective for the simulation of real-case scenarios. In the scope of this work a relaxed model can therefore not solve the real VRP but can serve as a basis

for further model improvement and specification. Tests for chosen algorithms were run with following settings to achieve consistent observations:

Algorithm	Number of iterations	Average calculation time (microseconds)
Random	1000	42126
NN	1	1580
Lin-Kernighan + Brute-Force	1	10091

Table 19 Comparison of time complexity

Each TSP route generated by compared algorithms was split into weekly trips and formed following VRP paths, each of them were calculated with the implementation that can be found in appendices B, C.

Algorithm	Path	Paths' length (km)	Total length (km)
Random	{[0, 6, 1, 10, 5, 8, 0], [0, 4, 2, 3, 9, 0], [0, 7, 0]}	[328.0606, 232.7485, 95.3552]	656.1643
NN	{[0, 8, 4, 6, 2, 3, 0], [0, 9, 1, 10, 5, 0], [0, 7, 0]}	[234.5517, 322.0256, 95.3552]	651.9325
Lin-Kernighan + Brute-Force	{[0, 7, 6, 2, 5, 10, 0], [0, 1, 3, 9, 4, 0], [0, 8, 0]}	[309.4935, 270.4311, 67.934]	647.8586

Table 20 Comparison of route optimality

Overall, proposed algorithm found the most optimal route for a cluster, outperforming NN algorithm by 0.62% and randomly generated and iteratively found shortest route by 1.27%.

Generating random routes and searching for the best permutation can be a useful heuristic to find a feasible solution. However, this approach may not always guarantee an optimal solution.

To test a bigger problem, tests were run for a cluster, formed with the previously selected K-means algorithm, which consists of 22 nodes and has its depot set as an address located in Munich. The following dictionary with keys for ID and values for cities was created. Addresses are intentionally not fully shown and represent only the city or district name, therefore names may show up multiple times.

- 0: 'Depot';
- 1: 'Regensburg';
- 2: 'Passau';
- 3: 'Bayreuth';
- 4: 'Nuremberg';
- 5: 'Munich';
- 6: 'Rosenheim';

- 7: 'Augsburg';
- 8: 'Weilheim';
- 9: 'Ingolstadt';
- 10: 'Landshut';
- 11: 'Munich';
- 12: 'Munich';
- 13: 'Fürstenfeldbruck';
- 14: 'Nuremberg';
- 15: 'Fürth';
- 16: 'Ingolstadt';
- 17: 'Rummelsberg';
- 18: 'Munich';
- 19: 'Pegnitz';
- 20: 'Cham';
- 21: 'Nuremberg';
- 22: 'Ismaning'.

These nodes and its geodata will be used further to tests chosen approaches for building a TSP route, as well as VRP paths after applying a Split procedure. From the table below it is clear that the TSP route construction is best performed by the last listed algorithm. It can be assumed that it can be also a good option to use in solving a VRP.

Route construction algorithm	TSP route length (km)	VRP paths	VRP route length (km)
Random + sort	1604.71	[0, 22, 6, 12, 11, 9, 0], [0, 1, 17, 7, 5, 13, 0], [0, 20, 2, 3, 16, 19, 0], [0, 14, 4, 15, 21, 8, 0], [0, 18, 10, 0]	2025.44
NN	1126.84	[0, 12, 5, 18, 11, 22, 0], [0, 13, 7, 16, 9, 1, 0], [0, 20, 10, 6, 8, 17, 0], [0, 14, 4, 21, 15, 19, 0], [0, 3, 2, 0]	1935.64
Lin-Kernighan + Brute-Force	851.09	[0, 22, 10, 2, 20, 1, 0], [0, 14, 21, 15, 3, 19, 0], [0, 16, 9, 17, 4, 7, 0], [0, 18, 11, 6, 8, 13, 0], [0, 5, 12, 0]	1359.8

Table 21 Comparison of approaches for a longer route

Despite the given results it cannot be claimed that the proposed algorithm will always result in the most optimal solution compared to other algorithms mentioned and listed above. Further tests might reveal that there is an even better solution possible. To test this assumption, we run tests for all clusters, found previously with the K-Means algorithms and find VRP routes, modelling a

weekly planning with five working days and two weekends. Each working day driver visits a single customer and performs a service. Throughout the work week, driver visits all customers, and returns to the depot at the weekend.

In the case of TSP, NN algorithm can simulate one way of manual route planning. It can be a simple and effective method for finding an initial tour. However, it is not guaranteed to generate an optimal tour. The quality of the solution produced by NN depends on the order in which the cities are visited, which can be influenced by the number of nodes and the starting location. Such approach can be used to construct a route for up to a few hundreds of nodes, above that limit it becomes computationally infeasible to compute the distances between all pairs of nodes. The NN approach was initially considered as not efficient enough to solve the VRP as it is alone not sufficient since it does not take into account the capacity constraints or time windows that are typically present in VRP instances.

A hybrid approach proposed here has shown good results, especially when applying the local optimization. LK algorithm is highly recommended for TSP problems of any size, because of the quality of the solution and time performance. An additional local optimization has improved the length of the resulting route.

The Split procedure is a heuristic commonly used in VRPs to convert a single tour, which represents the solution to the TSP, into multiple sub-tours that can be assigned to individual vehicles. This procedure involves identifying a set of nodes in the TSP tour that can serve as natural break-points to split the tour into multiple sub-tours. In our case, the tour was split to create paths representing weeks, including weekends when the driver does not make trips. These sub-tours are easier to manage and perform service for individual vehicles.

However, the Split procedure does not take into account the remoteness of the last node to the depot, which can lead to suboptimal solutions. To address this issue, we applied local optimization using enumeration. This approach can improve the sub-tours, but it may get stuck in a local optimum. A local optimum is a solution that is better than all nearby solutions, but not necessarily better than all possible solutions to the problem. In other words, it's a point in the search space where the objective function has the highest (or lowest) value in a particular neighbourhood of that point, but not necessarily the overall highest (or lowest) value.

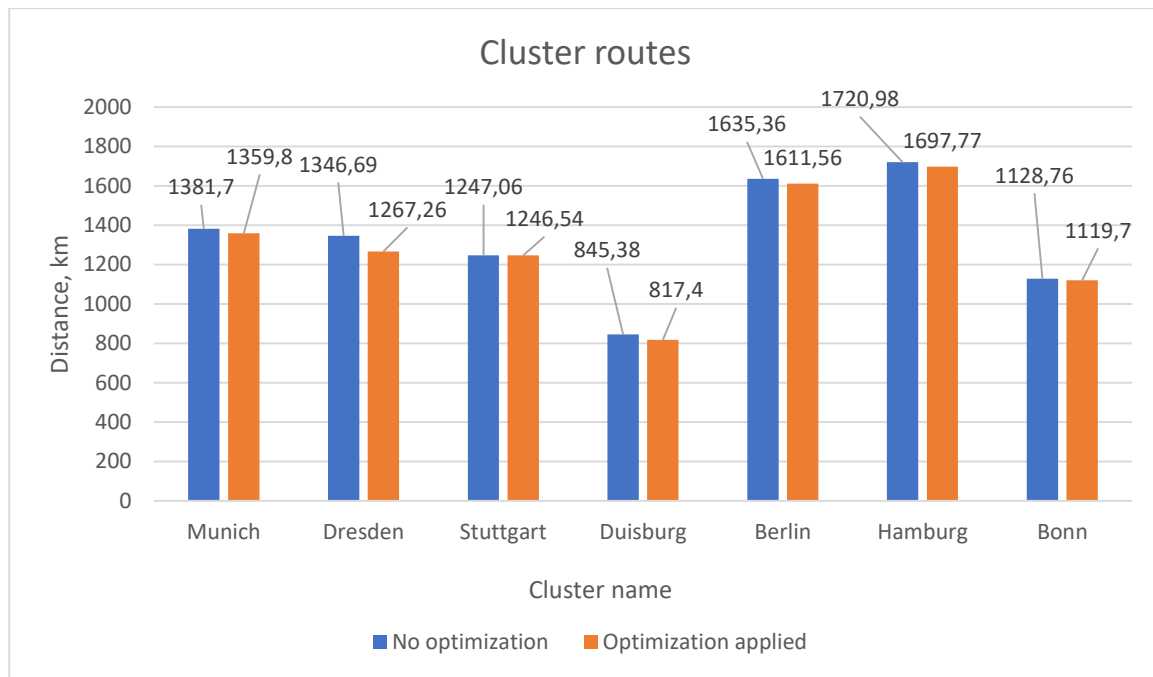


Figure 41 Algorithm applied to all cluster data

One way to show the difference between the results of a proposed hybrid approach and manual planning for solving VRPs is to compare the objective function values of the two approaches. The objective function is to minimize the total distance travelled by the vehicles and serves as a quantitative measure of the quality of a solution. It will be used to compare the efficiency and effectiveness of different planning methods. The lower the total distance is, the more efficient is the solution. An average route improvement for all clusters after local optimization with Lin-Kernighan approach reached 2.06% or 185.88 km overall shorter trips.

Cluster	Cluster size (nodes)	Random generation	KNN	LK+ local optimization
Munich	22	1880.66	1935.64	1359.8
Dresden	15	1305.79	1245.71	1267.25
Stuttgart	21	1438.48	1402.99	1246.54
Duisburg	32	1354.06	1134.986	817.4
Berlin	25	2047.17	1798.72	1607.93
Hamburg	26	2545.27	1895.1	1697.77
Bonn	13	1084.16	1211.957	1119.7

Table 22 Minimal routes found with different approaches

The proposed approach was able to outperform other algorithms in five cases out of seven to find an optimal VRP trip. The problem with local optima is that they can trap optimization algorithms and prevent them from finding the global optimum, which is the best possible solution to the problem. If an optimization algorithm gets stuck in a local optimum, it may not be able to escape and find a better solution. That can be noticed for single paths of the route – as they are constructed, they can be optimized only in scope of nodes they consist of, but not in scope of the whole set of nodes. This approach can be improved to avoid local optima by considering following techniques:

- Stochastic search algorithms: Stochastic search algorithms, such as genetic algorithms, Simulated Annealing, or Tabu Search, use randomized search strategies to explore the search

space more broadly and avoid getting trapped in local optima. These algorithms can be particularly effective when the search space is complex and has many local optima.

- **Multiple starting points:** Running the optimization algorithm from multiple starting points can help to increase the chances of finding a global optimum rather than getting trapped in a local optimum. This approach can be especially effective when there are multiple good solutions to the problem.
- **Neighbourhood search:** Neighbourhood search algorithms work by starting with a good solution and iteratively making small changes to the solution to explore nearby solutions. This approach can help to avoid local optima by gradually moving away from suboptimal solutions.
- **Problem-specific constraints:** Incorporating domain-specific constraints into the optimization problem can help to reduce the number of local optima and make the search space easier to navigate. For example, including constraints such as time windows or capacity limits can help to eliminate infeasible solutions and reduce the size of the search space.
- **Machine learning and data-driven methods:** Machine learning and data-driven methods can be used to learn from historical data and identify patterns in the data that can be used to guide the optimization process. For example, using machine learning to predict which customers are likely to order similar items or to predict traffic patterns can help to improve the efficiency of the optimization algorithm and avoid local optima.

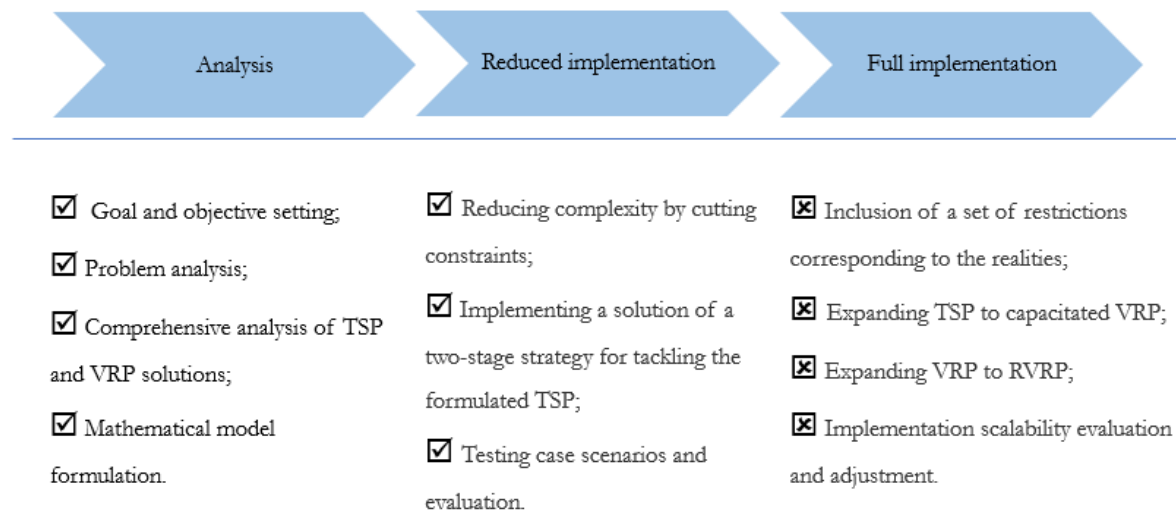
Lin-Kernighan algorithm is a more advanced method that has been shown to produce high-quality solutions for TSP. The algorithm is based on a local search technique that iteratively improves an initial solution by exchanging pairs of edges in a way that reduces the length of the tour. The algorithm has been widely used in the literature for most optimal TSP solution. In our case it has shown the best results to construct a TSP solution within a considerable time period and can be used in planning routes consisting of one day trips to different customers. In contrast to the Brute-Force algorithm, which takes hours to solve TSP for 13 nodes, it can be easily used to construct a route for hundreds of nodes.

In order to solve a VRP, Lin-Kernighan algorithm can be leveraged by conditional constraints, which would be incorporated into the search process. This can be done by modifying the cost function or by pre-computing certain costs using dynamic programming. With dynamic programming, the cost of certain subsets of the cities can be precomputed and these pre-computed costs can be used to guide the search for a better tour. The cost function in Lin-Kernighan is usually based on the distance between cities. However, it can be modified with additional constraints, such as time windows or capacity constraints. This can be done by adding penalty terms to the cost function, which increase the cost of violating the constraints.

In summary, while generating random routes and searching for the best permutation can be a useful heuristic for VRP, the Lin-Kernighan algorithm is a more advanced and reliable method that is likely to produce high-quality solutions.

5. Outcomes

The profitability of a business that provides business-to-business services is directly impacted by workforce planning and route building. These tasks are critical for eliminating unproductive time gaps and reducing fuel costs. The objective of this thesis was to identify opportunities for enhancing and automating route planning, with the implemented algorithm tested across various scenarios. While the implementation of the reduced VRP was successful, the capacitated multi-depot problem requires more extensive research and programming efforts.



This thesis focused on the VRP in the context of a healthcare services company, exploring the specific constraints and requirements of the industry. An objective function was developed and incorporated into a mathematical formulation to guide the analysis. The dataset provided by the company was formatted for a proper analysis. It revealed the need of a clustering method in order to group the data based on its similarity. With predefined cluster centres, all data points were assigned to a particular cluster with a K-Means method. It also shown to be useful in predicting further centroids in case the dataset will grow and there will be a need in a new depot. From the business view it is highly profitable as it simplifies and speeds up a hiring process for the HR and reduces overall driving costs.

Next a full and relaxed versions of a realistic VRP were developed and implemented. These could serve as a basis for the development of a two-phased algorithm. With the decomposition, problem was divided into subproblems, making it more tractable. It was attempted to apply a route first - cluster second approach to solve each single VRP problem – a big tour was first constructed with a Lin-Kernighan implementation and then locally optimized with an enumeration of all node sequences. This approach was compared to other two – NN and random paths generation and sorting approaches, which can be seen as a quick alternative to solve a relaxed VRP, but are unproductive for real VRP, where the complexity and search scope grows with a number of limitations.

We came to a conclusion that a Lin-Kernighan approach successfully solves middle and big size TSP and can be practically used for scheduling vehicle routes, considering constraints this specific problem might have. It can be used to solve the VRP as well, but first, it should be adapted to the specifics of the VRP by introducing additional constraints, such as vehicle capacity and time windows. This can be done by modifying the cost function used by Lin-Kernighan to take into account these constraints. For example, the cost function can include penalty terms for violating capacity constraints or time windows.

Another approach is to use Lin-Kernighan as a local search method within a larger metaheuristic framework, such as the Clarke and Wright savings algorithm or the Sweep algorithm. In this approach, Lin-Kernighan is used to improve the quality of the initial solution generated by the metaheuristic, by iteratively modifying the tour to decrease its length.

There are also other variations of Lin-Kernighan that have been specifically designed to solve the VRP, such as the Vehicle Routing Problem Lin-Kernighan Heuristic (VRP-LKH) and the Capacitated VRP Lin-Kernighan Heuristic (CVRP-LKH).

Overall, while Lin-Kernighan was originally developed for the TSP, it can be adapted to solve the VRP by introducing additional constraints or using it as a local search method within a larger metaheuristic framework.

References

- [1] Accenture, "Coronavirus: Supply Chain Disruption. Accenture Consulting," 16 March 2020. [Online]. Available: <https://www.accenture.com/be-en/insights/consulting/coronavirus-supply-chain-disruption>. [Accessed 2 November 2022].
- [2] G. Dantzig and J. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80-91, 1959.
- [3] M. M. Solomon and J. Desrosiers, "Survey paper—time window constrained routing and scheduling problems," *Transportation science*, vol. 22, no. 1, pp. 1-13, 1988.
- [4] G. Laporte, M. Gendreau, J. Y. Potvin and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International transactions in operational research*, vol. 7, no. 4-5, pp. 285-300, 2000.
- [5] Opex Analytics, "OPEX 101: Vehicle Routing Problems," Medium, 6 September 2019. [Online]. Available: <https://medium.com/opex-analytics/opex-101-vehicle-routing-problems-262a173f4214>. [Accessed 24 October 2022].
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to algorithms, MIT press., 2022.
- [7] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Computers & operations research*, vol. 34, no. 8, pp. 2403-2435, 2007.
- [8] H. R. Lourenço, O. C. Martin and T. Stützle, "Iterated local search," *Handbook of metaheuristics*, pp. 129-168, 2019.
- [9] G. Laporte, Y. Nobert and S. Taillefer, "Solving a family of multi-depot vehicle routing and location-routing problems," *Transportation science*, vol. 22, no. 3, pp. 161-172, 1988.
- [10] P. Fermin Cueto, I. Gjeroska and A. Sola Vilalta, "A solution approach for multi-trip vehicle routing problems with time windows, fleet sizing, and depot location," *Networks*, vol. 78, no. 4, pp. 503-522, 2021.
- [11] ADAC, "VW Caddy - Model Overview," ADAC Autokatalog, 2021. [Online]. Available: <https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/vw/vw-caddy/>. [Accessed 12 November 2022].
- [12] G. Laporte, P. Toth and D. Vigo, "Vehicle routing: historical perspective and recent contributions," *EURO Journal on Transportation and Logistics*, vol. 2, pp. 1-4, 2013.
- [13] C. Contardo and R. Martinelli, "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints," *Discrete Optimization*, vol. 12, pp. 129-146, 2014.

- [14] T. R. P. Ramos and I. G. Maria, "Solving a multi-product, multi-depot vehicle routing problem by a hybrid method.," in *Proc. of the 13th Portuguese Conf. on Artificial Intelligence (EPLA)*, Lisbon, Portugal, 2011.
- [15] T. Bulhoes, M. H. Ha, R. Martinelli and T. Vidal, "The vehicle routing problem with service level constraints," *European Journal of Operational Research*, vol. 265, no. 2, pp. 544-558, 2018.
- [16] G. Laporte, M. Desrochers and Y. Nobert, "Two exact algorithms for the distance-constrained vehicle routing problem," *Networks*, vol. 14, no. 1, pp. 161-172, 1984.
- [17] M. Dufay, C. Mathieu and H. Zhou, "An Approximation Algorithm for Distance-Constrained Vehicle Routing on Trees," *arXiv preprint arXiv:2210.03811*, 2022.
- [18] A. K. Garside and N. R. Laili, "A cluster-first route-second heuristic approach to solve the multi-trip periodic vehicle routing problem," *Jurnal Teknik Industri*, vol. 20, no. 2, pp. 172-181, 2019.
- [19] A. Choudhari, A. Ekbote and P. Chaudhuri, "Capacitated Vehicle Routing Problem Using Conventional and Approximation Method," *arXiv preprint arXiv:2208.00046*.
- [20] F. Meng, Y. Ding, W. Li and R. Guo, "Customer-oriented vehicle routing problem with environment consideration: two-phase optimization approach and heuristic solution," *Mathematical Problems in Engineering*, vol. 2019, pp. 1-15, 2019.
- [21] Y. Shi, L. Lv, F. Hu and Q. Han, "A heuristic solution method for multi-depot vehicle routing-based waste collection problems," *Applied Sciences*, vol. 10, no. 7, p. 2403, 2020.
- [22] Y. J. Gong, J. Zhang, O. Liu and R. Z. Huang, "Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 42, no. 2, pp. 264-267, 2011.
- [23] S. Geetha, G. Poonthalir and V. T. Vanathy, "A Hybrid Particle Swarm Optimization with Genetic Operator for Vehicle Routing Problem," *Journal of Advances in Information Technology*, vol. 1, no. 4, pp. 181-188, 2010.
- [24] W. Ho, G. T. Ho, P. Ji and H. C. Lau, "A hybrid genetic algorithm for the multi-depot vehicle routing problem," *Engineering applications of artificial intelligence*, vol. 21, no. 4, pp. 548-557, 2008.
- [25] J. Qian and R. Eglese, "Fuel emissions optimization in vehicle routing problems with time-varying speeds," *European Journal of Operational Research*, vol. 248, no. 3, pp. 840-848, 2016.
- [26] M. A. Islam, Y. Gajpal and T. Y. ElMekkawy, "Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 632-642, 2020.
- [27] L. Cassettari, M. Demartini, R. Mosca, R. Reve, R. Revetria and F. Tonelli, "A multi-stage algorithm for a capacitated vehicle routing problem with time constraints," *Algorithms*, vol. 11, no. 5, p. 69, 2018.
- [28] V. Lesch, M. König, S. Kounev and A. Stein, "Tackling the rich vehicle routing problem with nature-inspired algorithms," *Applied Intelligence*, vol. 52, no. 8, pp. 9476-9500, 2022.

- [29] J. Ramchandran and F. Marshall L., "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109-124, 1981.
- [30] T. Paolo and V. Daniele, "The vehicle routing problem," *Society for Industrial and Applied Mathematics*, 2002.
- [31] B. Golden, S. Raghavan and E. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*, New York, NY: Springer, 2008.
- [32] L. Du and R. He, "Combining nearest neighbor search with tabu search for large-scale vehicle routing problem," *Physics Procedia*, vol. 25, pp. 1536-1546, 2012.
- [33] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," *Computers & Industrial Engineering*, vol. 79, pp. 115-129, 2015.
- [34] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, no. 3, pp. 845-858, 2017.
- [35] J. E. Beasley, "Route first—cluster second methods for vehicle routing," *Omega*, vol. 11, no. 4, pp. 403-408, 1984.
- [36] C. Archetti and M. G. Speranza, "A survey on matheuristics for routing problems," *EURO Journal on Computational optimization*, vol. 2, no. 4, pp. 223-246, 2014.
- [37] W. Li, "A parallel multi-start search algorithm for dynamic traveling salesman problem," in *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10*, Springer, 2011, pp. 65-75.
- [38] H. K. Tsai, J. M. Yang, Y. F. Tsai and C. Y. Kao, "An evolutionary algorithm for large traveling salesman problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 4, pp. 1718-1729, 2004.
- [39] "Google OR-Tools documentation," Google Developers, [Online]. Available: <https://developers.google.com/optimization/>. [Accessed 25 November 2022].
- [40] A. Kumar and A. Munagekar, *Inventory, Storage and Routing Optimization with Homogenous Fleet in the Secondary Distribution Network Using a Hybrid VRP, Clustering and MIP Approach*, Springer, 2022, pp. 413-427.
- [41] A. Mahéo, "Implementing Lin-Kernighan in Python," Arthur Maheo, 2021. [Online]. Available: <https://arthur.maheo.net/implementing-lin-kernighan-in-python/>. [Accessed 8 December 2022].
- [42] N. Singh, "Traveling Salesman Problem (TSP) Implementation," GeeksForGeeks, 31 January 2023. [Online]. Available: <https://www.geeksforgeeks.org/traveling-salesman-problem-tsp-implementation/>. [Accessed 2 February 2023].
- [43] C. Prins, P. Lacomme and C. Prodhon, "Order-first split-second methods for vehicle routing problems," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 179-200, 2014.

- [44] K. C. Tan, L. H. Lee, Q. L. Zhu and K. Ou, "Heuristic methods for vehicle routing problem with time windows," *Artificial intelligence in Engineering*, vol. 15, no. 3, pp. 281-295, 2001.
- [45] T. Kanungo, D. M. Mount, N. S. Netanyahu, A. Y. Wu, R. Silverman and C. D. Piatko, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 881-892, 2002.
- [46] T. Ellinger, G. Beuermann and R. Leisten, *Operations research: eine Einführung*, Springer, 2013.
- [47] M. Mirabi, N. Shokri and A. Sadeghieh, "Modeling and solving the multi-depot vehicle routing problem with time window by considering the flexible end depot in each route," *International Journal of Supply and Operations Management*, vol. 3, no. 3, pp. 1373-1390, 2016.
- [48] R. He, W. Xu, J. Sun and B. Zu, "Balanced k-means algorithm for partitioning areas in large-scale vehicle routing problem," *Third International Symposium on Intelligent Information Technology Application*, vol. 3, pp. 87-90, 2009.
- [49] L. Schrage, "Formulation and structure of more complex/realistic routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 229-232, 1981.
- [50] R. Russell and W. Igo, "An assignment routing problem," *Networks*, vol. 9, no. 1, pp. 1-17, 1979.
- [51] A. Pessoa, R. Sadykov, E. Uchoa and F. Vanderbeck, "A generic exact solver for vehicle routing and related problems," *Mathematical Programming*, vol. 183, pp. 483-523, 2020.
- [52] K. Helsgaun, "General k-opt submoves for the Lin–Kernighan TSP heuristic," *Mathematical Programming Computation*, vol. 1, pp. 119-163, 2009.
- [53] F. Cavaliere, E. Bendotti and M. Fischetti, "An integrated local-search/set-partitioning refinement heuristic for the Capacitated Vehicle Routing Problem," *Mathematical Programming Computation*, vol. 14, no. 4, pp. 749-779, 2022.
- [54] C. Prins, N. Labadi and M. Reghioui, "Tour splitting algorithms for vehicle routing problems," *International Journal of Production Research*, vol. 47, no. 2, pp. 507-535, 2009.
- [55] C. Archetti, M. W. P. Savelsbergh and M. G. Speranza, "Worst-case analysis for split delivery vehicle routing problems," *Transportation science*, vol. 40, no. 2, pp. 226-234, 2006.
- [56] C. Jean-François, L. Gilbert, S. Martin and V. Daniele, "Vehicle Routing," in *Transportation*, Elsevier, 2007, pp. 195-224.
- [57] Y.-M. Shen and R.-M. Chen, "Optimal multi-depot location decision using particle swarm optimization," *Advances in Mechanical Engineering*, vol. 9, no. 8, 2917.
- [58] J. Sajaykumar, "Vehicle Routing Problem and Its Variants," 3 July 2019. [Online]. Available: <https://www.linkedin.com/pulse/vehicle-routing-problem-its-variants-sajaykumar-j/>. [Accessed 26 October 2022].

-
- [59] R. Gour, "Heuristic Search in Artificial Intelligence - Python," 5 October 2018. [Online]. Available: <https://medium.com/@rinu.gour123/heuristic-search-in-artificial-intelligence-python-3087ecfece4d>. [Accessed 2 November 2022].
- [60] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European journal of operational research*, vol. 126, no. 1, pp. 106-130, 2000.
- [61] A. Jungwirth, M. Frey and R. Kolisch, The vehicle routing problem with time windows, flexible service locations and time-dependent location capacity, 2020.

Appendices

Appendix A: TSP Solver

```
from sys import maxsize
from itertools import permutations
from datetime import datetime

time_arr = []
# function to implement TSP
def TSPSolution(graph, s, cities):

    # keep all vertex other than the starting point
    vertex = []

    # traverse the diagram
    for i in range(cities):
        if i != s:
            vertex.append(i)

    # keep minimum weight
    min_path = maxsize

    next_permutation = permutations(vertex)
    best_path = []

    for i in next_permutation:
        # store current Path weight(cost)
        current_pathweight = 0

        # compute current path weight
        k = s

        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]

        # update minimum
        if current_pathweight < min_path:
            min_path = current_pathweight
            best_path = [s]
            best_path.extend(list(i))
            best_path.append(s)

    return min_path, best_path
```

Listing A. 1 Algorithm to find TSP based on Brute-Force approach

Appendix B: KNN Algorithm

```
import random
def get_neighbors(cluster,city_name):
    distances = []
    indexes = []
    indexes.append(0)
    matrix = addDepot(cluster, city_name) # function to add the depot and create a distance matrix

    for row in matrix:
        row_values = matrix["point_"+str(indexes[-1]+1)]
        sorted_row = sorted(row_values)
        row_list = row_values.tolist()

        for i in range(len(row_list)): #finding the smallest weight in the matrix
            second_smallest = sorted_row[i]
            second_smallest_index =row_list.index(second_smallest)
            if(second_smallest_index not in indexes and i>0):

                dist = row_values["point_"+str(second_smallest_index+1)]

                distances.append(dist)
                indexes.append(second_smallest_index)
                break
    distances.append(matrix["point_"+str(indexes[-1]+1)]["point_1"])
    indexes.append(0)

    k=0
    p=6
    sum_arr = []
    while k < len(indexes): # create an array of indexes
        arr = indexes[k:p]
        k=p
        p+=5
        sum_arr.append(arr)

    for i in sum_arr: # adding 0 to indicate a depot
        if i[0]!=0:
            i.insert(0,0)
        if i[-1]!=0:
            i.append(0)

    cluster = {}
    cl_key = 0 # create a dictionary to assign paths to weeks
    for i in range(1,len(sum_arr)):
        cluster[cl_key] = sum_arr
        cl_key+=1
```

```

NN_paths_total = calcAll(cluster,matrix) # function to calculate total distance of
the trip
return sum(NN_paths_total) #Overall paths' distances sum

```

Listing B. 1 Algorithm for finding NN

Appendix C: Random VRP Path Generation

```

def findShortestRand(n, city, city_name):
    results = []
    # adding a depot to build a distance matrix
    depot = coaches.loc[coaches["City"]==city_name]

    d_list = depot[["Latitude","Longitude"]].values
    d_list = d_list[0].flatten().tolist()

    lat = d_list[0]
    lon = d_list[1]
    lat = np.float64(lat)
    lon = np.float64(lon)
    new_row = pd.DataFrame({'latitude': [lat], 'longitude': [lon]})
    city = pd.concat([new_row, city]).reset_index(drop=True)
    matrix =createDistanceMatrix(city)

    #finding the best route in 1000 iterations
    for i in range(1000):
        path = genRandpaths(n)
        result = calcAll(path, matrix)
        results.append([path, result])

    for i in range(len(results)):

        results[i][1] = sum(results[i][1])

    sorted_results = sorted(results, key=lambda x: x[1])
    return sorted_results[0]

#generating VRP paths representing weeks
def genRandpaths(n):
    rand_path = genRan(n)
    k=0
    p=6
    sum_arr = []
    while k < len(rand_path)-1 :
        arr = rand_path[k:p]
        k=p
        p+=5
        sum_arr.append(arr)

    for i in sum_arr:

```

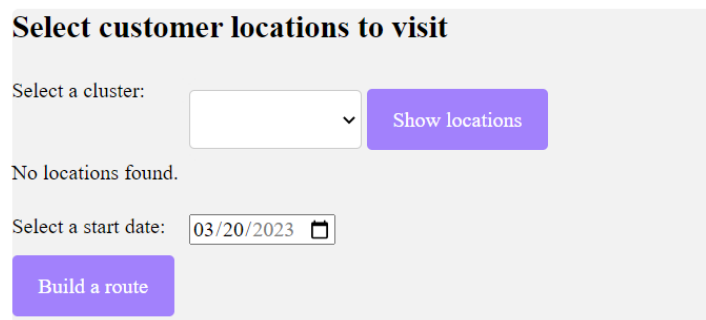


```
if i[0]!=0:
    i.insert(0,0)
if i[-1]!=0:
    i.append(0)
# returning an array of VRP paths
return sum_arr
```

Listing C. 1 Algorithm for random path generation

Appendix D: UI of the Web Application

The application can be started from the desktop with the path to the .bat file which starts the main .py programm, so that the employee does not need to start the file from the command-line interface.



Select customer locations to visit

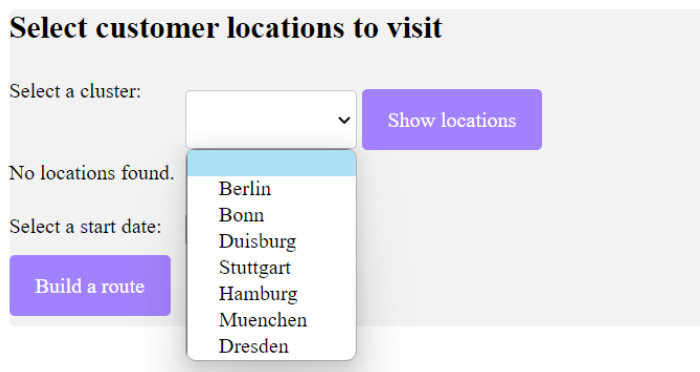
Select a cluster: Show locations

No locations found.

Select a start date: 03/20/2023

Build a route

Figure D. 1 Start screen



Select customer locations to visit

Select a cluster: Show locations

No locations found.

Select a start date:

Build a route

- Berlin
- Bonn
- Duisburg
- Stuttgart
- Hamburg
- Muenchen
- Dresden

Figure D. 2 Choosing a cluster from the list of available depots

Select customer locations to visit

Select a cluster:

Select All

- Neubrandenburg
- Berlin
- Potsdam
- Cottbus
- Meißen
- Dresden
- Bautzen
- Berlin
- Dresden
- Dresden
- Cottbus
- Sommerfeld
- Templin
- Rügen
- Berlin
- Hoyerswerda
- Dresden
- Senftenberg
- Lauchhammer
- Klettwitz
- Freiberg
- Berlin
- Königs-Wusterhausen
- Lübben
- Woltersdorf

March 2023 ↑ ↓

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Clear Today

Select a start date:

Figure D. 3 Choosing nodes and a start date of the route