universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Solving a Real Data Waste Collection VRP using Google OR-Tools

verfasst von | submitted by

Katarina Đajić

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 915
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von | Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner Privatdoz.
Mitbetreut von | Co-Supervisor:	Alina-Gabriela Müller BSc MSc PhD

Abstract (German)

Diese Masterarbeit bewertet die Fähigkeiten von Google OR-Tools bei der Lösung realer Vehicle Routing Problem anhand einer Fallstudie zur kommunalen Abfallsammlung in Sarajevo, der Hauptstadt von Bosnien und Herzegowina. Die Ineffizienz bei der Zuweisung von Fahrzeugen zu Routen und das wachsende Abfallaufkommen machten einen systematischen Ansatz zur Verbesserung der betrieblichen Effizienz und Nachhaltigkeit erforderlich. Die Studie konzentriert sich auf die Minimierung der Gesamtfahrstrecken für die Sammlung von Einwegabfällen, die in gemischten Abfallbehältern entsorgt werden.

Da einige Behälter täglich, andere zweimal oder dreimal pro Woche geleert werden, wird das Problem als Periodic Vehicle Routing Problem (PVRP) über einen wöchentlichen Zeithorizont formuliert. Diese periodische Natur des Problems erfordert einen zweistufigen Ansatz. In der ersten Phase wird die heuristische Methode Large Neighborhood Search (LNS) angewendet, um die Container den Tagen im wöchentlichen Planungszeitraum zuzuordnen. In der zweiten Phase werden die täglichen Routen mit OR-Tools, einem Open-Source-Optimierungstool, generiert. Unter Verwendung realer Geodaten, die über die OpenRouteService-API bezogen wurden, berechnet die Arbeit realistische Fahrstrecken, und dieselbe API wird zur Visualisierung der Routen verwendet. Das primäre Ziel des Problems ist die Minimierung der Gesamtfahrstrecken.

Python wird für die Datenverarbeitung, Modellimplementierung und Ergebnisvisualisierung verwendet, während die Openrouteservice-API für die interaktive Routenpräsentation eingesetzt wird. Die Studie bewertet verschiedene Algorithmuslösungen für unterschiedliche Problemgrößen und analysiert Faktoren wie Rechenzeit und Gesamtfahrstrecke.

Die Ergebnisse zeigen, dass der Solver bei Benchmark-Instances eine gute Leistung erbracht hat, indem er in mehreren Fällen die besten bekannten Lösungen (BKS) erzielte und bei allen ausgewählten Instances innerhalb von 10 % der BKS blieb. In Bezug auf den realen Fall hat der Solver innerhalb eines angemessenen Zeitrahmens erfolgreich realisierbare Routen generiert. Um die Qualität der Lösung jedoch besser beurteilen zu können, ist ein Vergleich mit dem bestehenden Routenplan erforderlich,

um zu bewerten, ob die Fahrstrecke verkürzt oder die Fahrzeugauslastung verbessert wurde. Insgesamt zeigen die Ergebnisse, dass OR-Tools als Open-Source-Optimierungstool den Entscheidungsprozess effektiv unterstützen und eine kostengünstige Alternative zu kommerziellen Softwarelizenzen darstellen kann, was zu effizienteren Abfallsammelplänen beiträgt, die Kosten senken und die Ressourcennutzung verbessern.

Abstract (English)

This master's thesis evaluates the capabilities of Google OR-Tools in solving real-life vehicle routing problems, using municipal waste collection in Sarajevo, the capital of Bosnia and Herzegovina, as a case study. The inefficiency in assigning vehicles to routes and the growing waste volume raised a need for a systematic approach to improve operational efficiency and sustainability. The study focuses on minimizing total travel distances of routes for the collection of a single-type waste deposited in mixed-waste containers.

Given that some containers are emptied every day, others twice or three times a week, the problem is formulated as a Periodic Vehicle Routing Problem (PVRP) over a weekly time horizon. This periodic nature of the problem requires a two-phase approach. In the first phase Large Neighborhood Search (LNS) heuristic is applied to assign containers to days across the weekly planning horizon. In the second phase, the daily routes are generated using OR-Tools, an open-source optimization tool. Using real-world geospatial data obtained via the OpenRouteService API, the thesis calculates realistic driving distances, and the same API is used to visualize the routes. The primary objective of the problem is to minimize total travel distances.

Python is used for data processing, model implementation, and result visualization, while OpenRouteService API is used for interactive route presentation. The research evaluates different algorithm solutions across varying problem sizes, analyzing factors such as computational time and total travel distance.

The results indicate that the solver demonstrated good performance on benchmark instances, achieving best-known (BKS) solutions in several instances, and remaining within 10% of the BKS across all selected instances. Regarding real-life case, the solver successfully generated feasible routes within a reasonable time frame. However, to assess the quality of the solution more, comparison with the existing route plan to evaluate whether travel distance is reduced or vehicle utilization is improved is required. Overall, the findings show that OR-Tools, as an open-source optimization tool, can effectively support decision-making process and provide a cost-efficient alternative to commercial software licenses, contributing to the more efficient waste collection plans that reduce costs and improve resource utilization.

Acknowledgments

First and foremost, I would like to thank Mrs. Alina-Gabriela Müller, BSc MSc PhD, for her guidance, valuable tips, and encouragement throughout the writing of this thesis. Her expertise and advice were essential in navigating the challenges persisted, and her willingness to share knowledge and provide direction played a crucial role in its completion. Without all insightful comments and constructive suggestions this thesis would not have reached its final form.

I am also grateful to Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privatdoz. for always being open to discussion and for offering thoughtful and constructive feedback. His readiness to engage with my ideas and proposals has been an important source of motivation during this process.

Finally, I wish to express my deepest appreciation to my family and my boyfriend Aleksandar for their continuous support and belief in me, even at times when I struggled to believe in myself.

I List of Abbreviations

VRP	Vehicle Routing Problem
CVRP	Capacitated Vehicle Routing Problem
PVRP	Periodic Vehicle Routing Problem
MTPVRP	Multi-Trip Periodic Vehicle Routing Problem
LNS	Large Neighborhood Search
TS	Tabu Search
LS	Local Search
GLS	Guided Local Search
SA	Simulated Annealing
KJKP “Rad”	Cantonal Public Utility Company “Rad”
TSP	Traveling Salesman Problem
MSP	Minimum Spanning Tree
MMM	Minimum Maximal Matching
ORS	Open Route Service

II List of Figures

Figure 1 Seven municipalities in Sarajevo Canton 17

Figure 2 Collection Points in the Municipality of Novo Sarajevo 18

Figure 3 Scheduling Phase Pseudo Code 35

Figure 4 Pseudo Code for solving the Multi-Trip Routing Problem with OR-Tools ... 48

Figure 5 The City Map with Macro-Points 54

Figure 6 Route Visualization for Day 1 57

Figure 7 Route Visualization for Day 2 58

Figure 8 Route Visualization for Day 3 59

Figure 9 Route Visualization for Day 4 60

Figure 10 Route Visualization for Day 5 61

Figure 11 Route Visualization for Day 6 62

Figure 12 Route Visualization for Day 7 63

Figure 13 Time-Limit Parameter Tuning 64

III List of Tables

Table 1 Input Data File Structure 50

Table 2 Selected Instances Characteristics 51

Table 3 Comparison with Best Known Solutions 53

Table 4 Real-Life Instance Characteristics 55

Table 5 Real-Life Input File Structure 55

Table 6 Execution Summary for the 7 days Planning Horizon 56

Table 7 Day 1 Routing Performance 57

Table 8 Day 2 Routing Performance 58

Table 9 Day 3 Routing Performance 59

Table 10 Day 4 Routing Performance 60

Table 11 Day 5 Routing Performance 61

Table 12 Day 6 Routing Performance 62

Table 13 Day 7 Routing Performance 63

1. Introduction

Considering the growing waste management challenges worldwide, including global pollution, resource depletion, and increasing waste volumes, establishing an efficient waste management model in the city of Sarajevo becomes especially significant and aims to serve as an example that could inspire other cities in Bosnia and Herzegovina.

The objective of this approach is not only to contribute to environmental preservation but also to provide considerable economic benefits, such as reduced waste collection costs, improved resource efficiency, and fostering innovation in production and technology.

In the strategic plan of KJKP RAD, the vision of the company is clearly defined: to become a leading regional enterprise in integrated, sustainable, modern, environmentally oriented, and economically efficient waste management (KJKP “RAD” d.o.o. Sarajevo (2022). Guided by the mission of ensuring a clean and beautiful Sarajevo, the aim is to continuously strive to improve services and infrastructure. Building on this foundation, our contribution in this research will be to optimize waste collection routes, which will not only reduce costs but also benefit the environment.

In the capital of Bosnia and Herzegovina, significant waste management challenges have persisted for a long time. There are multiple reasons for this, with some of the most prominent being the insufficient number of waste disposal containers, many defective containers, an aging vehicle fleet, and an increase in overall waste volume.

With the development of a local waste management plan, the introduction of an additional vehicle fleet, there has arisen a need for more efficient coordination. Properly collected waste will significantly ease the workload of municipal workers, allowing them to shorten the operations time, leaving them time for additional responsibilities such as recycled waste sorting.

There is no established method for assigning vehicles to routes. Instead, this task is performed according to the current situation and the available number of garbage trucks. The goal of the thesis is to contribute by optimizing routes for municipal vehicles to minimize total travel distances. This approach not only reduces operational costs but also supports environmental sustainability, improves resource efficiency, and

fosters long-term sustainable waste management in the capital of Bosnia and Herzegovina.

This study aims to evaluate whether the optimization problem can be efficiently solved using Google's OR-Tools, an open-source software specifically designed for optimizing vehicle routing problems and other optimization challenges.

The remaining work is as follows: Chapter 2 is composed of a literature review on urban waste collection optimization, a summary of waste management vehicle routing problem solution methods and heuristics to solve this type of problem, along with an introduction to Google's OR-Tools. Chapter 3 provides a detailed description of the problem, offering an overview of the current waste collection situation in Sarajevo. Google OR-Tools algorithms and OpenRouteService platform are discussed in chapters 4 and 5, respectively. Solution approach is described in detail in chapter 6, while simulations in OR-Tools and visual representations are shown in chapter 7. Finally, Chapter 8 presents current limitations and chapter 9 concludes the thesis with a summary, key takeaways and possible ideas for future research.

2. Vehicle Routing Problem

The second part of the paper presents a literature review that begins with an introduction to vehicle routing problems, followed by an overview of recent and relevant studies on waste collection VRPs and VRPs that specifically address our problem.

2.1 Vehicle Routing Problem

One of the most researched optimization problems is the vehicle routing problem (VRP). It belongs to the NP hard type of problem, as no exact algorithm can be applied to the scope and instances related to real-life problems. As stated by Toth and Vigo (2002), VRP can be described as a problem related to finding the optimal solution or the optimal set of routes for a fleet of vehicles to serve a certain number of customers or collection nodes, considering many restrictions, which are determined depending on the objective problem. It is considered as one of the most studied and significant combinatorial optimization problems. The goal of VRP is actually to determine a set of optimal routes for a certain number of vehicles that move from the depot as starting points to customers, and return to the depot, while the limitations such as vehicle capacity, or time windows, must be satisfied, and costs must be minimized. According to Belmabrouk, Lahmar, Chouikhi, and Bentaher (2022) transportation is one of the main contributors to CO2 emissions, positioning it among other sectors as most harmful to the environment and air quality. Within this context, VRP represents a highly relevant sub-problem, where it is important to be aware that optimal routes are usually most cost-efficient ones, but that does not necessarily mean the shortest paths, as in real-life scenarios multiple factors and constraints must be considered in routes design. Hence, this paper aims to solve VRP in the field of waste management, with a particular focus on urban waste collection.

2.2 Vehicle Routing Problem with Particular Focus on Waste Collection

In a waste collection problem, utility companies must take waste from the garbage collection points and transport it to dedicated treatment facilities (Han & Ponce-Cueto, 2015). Similar to the classic VRP, where a fleet of vehicles is assigned to different tasks and the primary objective is to optimize the routes to minimize costs such as time, distance, or fuel consumption, waste collection can also be considered as a variant of this problem.

At its core, waste collection as a process can be formulated as a Vehicle Routing Problem, with the purpose of determining how a fleet of vehicles can efficiently serve a set of customers or collection points. A typical waste collection VRP includes vehicles, a depot, which serves a start and an end point, and a number of collection points to be visited. The complexity of the problem increases when additional constraints are introduced, such as heterogenous fleets, multiple disposal facilities or specific operational constraints (Beliën, De Boeck, & Van Ackere, 2012). Typical constraints considered in this type of a problem usually include periodicity, vehicle capacity, and workload balancing.

There are a number of studies that have focused on developing a variety of heuristic and metaheuristic methods and algorithms in order to find a feasible solution for efficient waste collection.

One of the first research on waste collection and its routing challenges was carried out by Beltrami and Bodin (1974), who studied the cases of Washington and New York City using a group of vehicles. Building on the pioneer savings heuristic introduced by Clarke and Wright in 1964, they adapted the method to determine the best routes. Their approach was widely adopted by municipalities and proved to be highly efficient.

After this first application of VRP to waste collection in the 1970s (Beltrami & Bodin, 1974), literature has expanded considerably over the years, addressing increasingly complex real-life requirements. Recently, Hess et al. (2024) provided a comprehensive survey on routing aspects of waste collection, where the most important papers are grouped by problem type and discussed in connection with their common problem characteristics. Furthermore, their work identifies research directions that go beyond the usual routing objectives, including the use of sensors, adapting replenishment routes to also include waste collection for more sustainability, and the use of hybrid or electric vehicles. The survey shows the current state of the art in waste collection routing and underlines the opportunities for future research.

To respond to frequent changes and factors, it becomes important to explore approaches that can generate new routes on a regular basis. In addition, advanced

algorithms trained on real-life cases should be considered, as they are more suitable for addressing today's waste collection challenges.

Since OR-Tools in its improvement solution strategy considers Tabu Search and Guided Local Search, several studies that demonstrate the superiority of these algorithms for complex VRPs in waste management are reviewed here. For example, Gómez et al. (2015) applied Tabu Search within the framework of Multi-Objective Adaptive Memory Programming to minimize the transport costs and improve the service levels in a real-life environment. Molina et al. (2019) developed a variable neighborhood Tabu Search for the municipality of Seville, where results improved upon the solutions used by the local authorities. Wenhua et al. (2015) proposed an improved Tabu Search for recycling waste household appliances, with objectives of maximizing resource recycling and minimizing recycling loss, showing promising results.

Guided Local Search (GLS) is another metaheuristic that works by balancing cost and penalties, incorporating features of both Tabu Search and Simulated Annealing. Barbucha et al. (2011) introduced an agent based GLS algorithm combined with an asynchronous team concept to solve the capacitated VRP.

Building on these algorithms, recent studies have applied Google OR-Tools, to develop and test solution approaches for waste collection problems. Dowd, Dixon and Kinsella (2020), showed that GLS as an improvement algorithm of OR-Tools together with other open tools such as OpenStreetMap can improve household sanitation services in urban Haiti zones, reducing costs and fuel consumption. Similarly, Silva et al. (2023), applied GLS in case study of Braganca (Portugal), where cost comparison between real and optimized scenarios highlighted significant savings. Comparable findings are obtained from Algethami (2023), where GLS was used to minimize fuel consumption and the number of trucks, improving the total route design process for waste management in Saudi Arabia. Finally, Dereci and Karabekmez (2022) presented a case study of Umraniye district in Istanbul, where OR-Tools were tested with the real waste collection data, again confirming the potential of modern and dynamic optimization methods.

2.3 Periodic Vehicle Routing Problem

Beltrami and Bodin (1974) were among the first ones to introduce an approach to vehicle routing problems in the context of waste collection, demonstrating not only perspective of waste collection, but also one of the earliest formulations of Periodic Vehicle Routing Problems. They presented an innovative approach for managing garbage collection in New York and since then the problem has been extended to numerous variations, including various additional constraints, among others multiple depots, multiple trips and heterogenous vehicles.

The PVRP is the routing problem which is highly relevant in real-world applications. It is introduced often naturally, as it arises in context where companies must perform periodic repair or maintenance activities, or where goods need to be collected or transported following a specific recurring schedule. Urban logistics is another important area, where strict regulations or infrastructure limitations often prohibit the use of extremely large vehicles, making periodic routing strategies even more important.

Recent studies highlight the continuous innovations in this domain of the VRP, especially with the focus on the multi-trip approach. For instance, Bernardino et al. (2022) studied a Multi-Trip PVRP with release dates and interrelated periods in the context of a car components distribution company, applying a metaheuristic based on a rolling-horizon approach. Similarly, Garside and Laili (2019), proposed a cluster-first, route-second heuristic for a petroleum gas distributor. Mahdavi, Mansour and Sajadieh (2021), developed a model that includes intermediate transfer stations, making the problem multi-trip specific, applying it to a case study of the city of Tehran. Another real-life application is done by Trikolaee et al. (2019) introducing a novel simulated annealing heuristic to address the MTPVRP, validated through a case study conducted in the city of Sanandaj.

As established in the literature, no heuristic or metaheuristic method has consistently demonstrated the ability to solve the waste collection problem to optimality, given its inherent complexity and the number of constraints involved. This paper aims to evaluate the effectiveness of Google's OR-Tools and its integrated heuristic and metaheuristic approaches in addressing this challenging problem. The primary focus will be on assessing whether these methodologies can produce feasible solutions that satisfy all constraints and, ideally, approach or achieve optimality. By analyzing the

performance and outcomes of these techniques, this study seeks to contribute to the ongoing exploration of computational methods for solving real-world waste collection problems.

3. Problem Description

Municipal waste collection refers to the collection and transport of waste from the numerous waste collection points to the place of treatment or discharge (such as a landfill or a recycling facility) that is managed by, or on behalf of, municipalities. According to UN-Habitat (2010), municipal waste collection systems often involve high operational costs and can represent a significant portion of the overall municipal budget. Therefore, optimization of the waste collection process can create potential cost savings for municipalities.

The city of Sarajevo seeks to improve the efficiency of its waste collection system, which currently relies on more than 10.000 containers distributed across the Sarajevo Canton (Municipality of Novo Sarajevo, 2016-2021). These containers are serviced on a fixed schedule, with collection frequencies either every day or two or three times per week. This existing waste collection system serves as a case study to examine whether an optimization-based approach, implemented by Google-OR Tools, can be applied to a real-life routing problem.

The study was conducted independently, without direct collaboration or communication with Sarajevo's public utility company. Consequently, due to limited access to real-life data, several aspects of the problem were approximated, and the overall complexity of the system was simplified. These modifications were necessary to define clear requirements and generate actionable results within the constraints of the study. Despite these limitations, the research aims to provide valuable insights into the potential application of advanced optimization tools for waste management in Sarajevo.

Waste management in the municipalities of the Sarajevo Canton is specific compared to other municipalities in Bosnia and Herzegovina. Sarajevo Canton is divided into nine districts, where the collection, transport, and disposal of municipal waste from residential and commercial buildings, as well as commercial spaces, have been assigned to the jurisdiction of the Sarajevo Canton. The Canton has entrusted these tasks to the Cantonal Public Utility Company “Rad” (Municipality of Novo Sarajevo, 2016-2021).

Although this creates a specific situation regarding the municipality's role within the broader solid waste management system, the Municipality of Novo Sarajevo, on of the nine municipalities in the Canton, developed a dedicated Waste Management Plan, which serves as a foundation for our research. Within its authority and in cooperation with KJKP “Rad”, the municipality aims to improve the current situation concerning waste reduction, separate collection, and transportation of municipal waste from its territory.

The seven municipalities in Sarajevo Canton are shown in Figure 1. However, our research focuses solely on a residual, mixed-waste collection in the Municipality of Novo Sarajevo. A visual representation of all container locations in the Novo Sarajevo area is shown in Figure 2. This approach aligns with the current distribution of responsibilities, as waste collection services are already organized by districts but managed by a single public company. Each garbage vehicle collects only a single type of waste. The distinction between different districts and different types of waste divides the problem into isolated subproblems, where our subproblem contains close to 500 waste containers. The paper focuses only on residual waste since having a large enough dataset is one of the prerequisites to adequately test the performance of Google OR-Tools on large and complex VRPs, and not enough data for compostable waste is currently available.

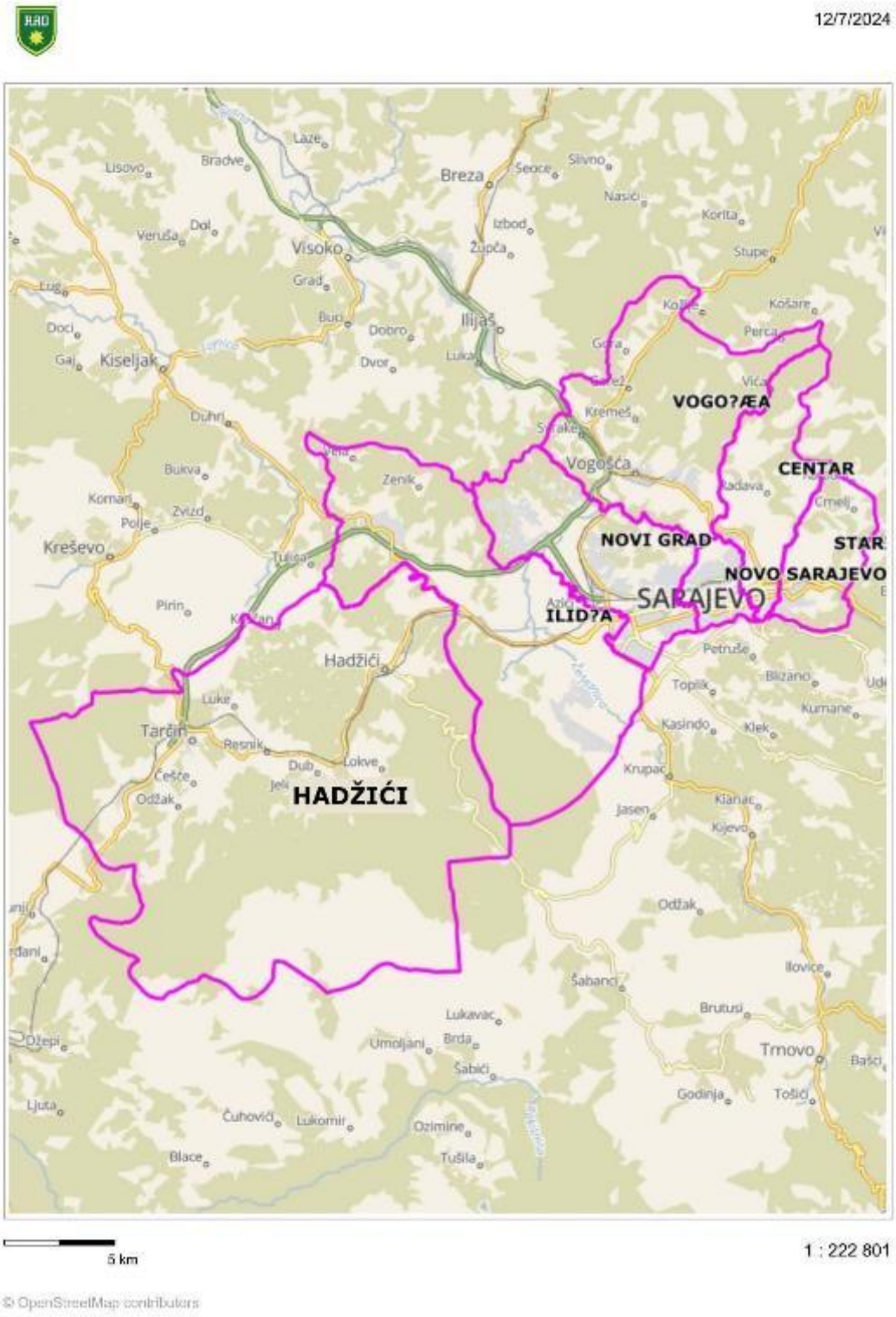

Figure 1 Seven municipalities in Sarajevo Canton

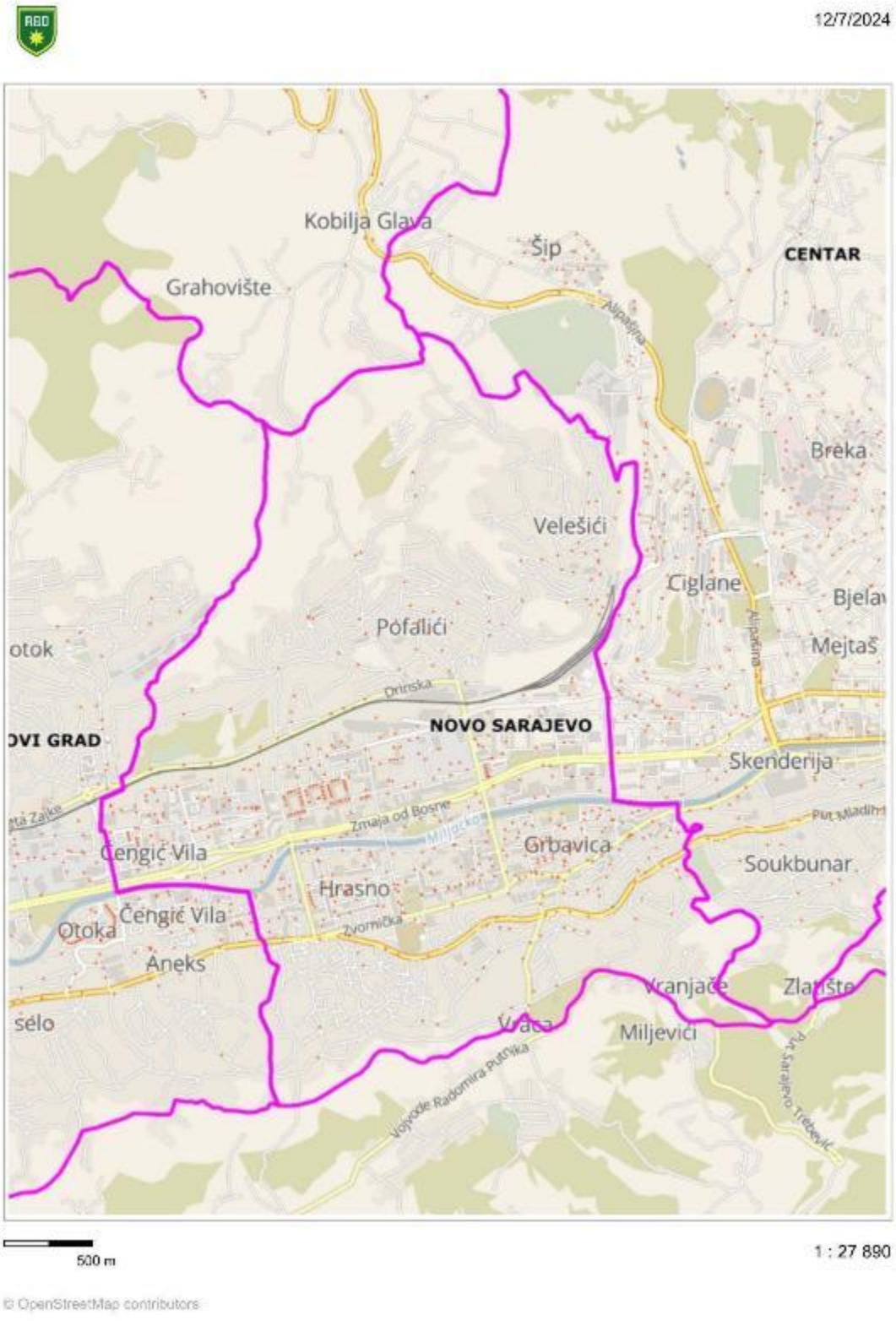

Figure 2 Collection Points in the Municipality of Novo Sarajevo

The Municipality of Novo Sarajevo belongs to the Sarajevo Canton and covers an area of 9.9 km². The distance between the northernmost and southernmost points of the municipality is 4.5 km, while the distance between its westernmost and easternmost points is 3.75 km (Municipality of Novo Sarajevo, 2016-2021). The Smiljevići landfill serves as the central processing facility for mixed waste. Within the area of the Municipality of Novo Sarajevo, mixed waste containers are available in three different sizes: 1100, 2500, and 3000 liters. The number of containers available is 497. Six specialized garbage collection vehicles with capacities ranging from 7,1 to 7,8 tonnes are used for waste collection operations. However, for simplification in our research, a homogenous vehicle capacity of 7,3 tonnes is assumed. This research addresses a multi-trip VRP, where each vehicle is restricted to a maximum of four routes per day.

We do not have access to historical data regarding the exact fill levels of the containers. Although we have average values, the precise amount of waste at any specific time for individual containers is unknown. Consequently, the exact fill rates for containers are also unknown. Given that the municipality faces challenges with container availability, the fill rate for each container is defined as full container capacity, assuming they are always filled to capacity at each collection.

Containers are serviced either two, three, or seven days per week. The formulation of the problem considers that waste collection personnel operate in one shift daily during nighttime hours. In our model, breaks are not considered, and garbage trucks are restricted from making multiple trips within a single shift to collect additional waste.

The map provided by KJKP “RAD” indicates the locations of all containers, as well as the container type at each site, specifying the type of waste for which each container is intended. Additionally, it highlights the capacity of each container and the scheduled day for emptying.

This study addresses a capacitated multi-trip VRP without time windows, meaning customers do not require service within specified time intervals. The only operational limitation is that vehicles can make a maximum of five trips a day. Furthermore, drivers break is not included, and service times are represented as zero across all locations.

In our research, we aim to optimize garbage collection operations over a recurring weekly schedule, aligning with the operational patterns of garbage collection in the district. This weekly focus allows us to analyze how collection routes can be designed to minimize travel distances and improve efficiency, given the constraints of single shifts and fixed truck capacities. By defining the problem as a Periodic Vehicle Routing Problem (PVRP), we account for the fact that containers are not serviced daily but follow specific schedules (three, six, or seven visits per week). This periodic structure ensures that waste collection is balanced across the week while adhering to operational constraints.

Our analysis evaluates the connection between container capacities, collection frequencies, and route optimization to propose a strategy that minimizes operational costs while maintaining service reliability.

4. Google's Operations Research (OR)-Tools

OR-Tools, developed by Google, is a powerful open-source optimization tool widely known for its robust capabilities in solving complex optimization problems (Cuvelier et al., 2023). Its advantage lies in its advanced implementation of constraint programming, which is particularly advantageous for addressing Vehicle Routing Problems (VRPs). As VRPs often involve large-scale, complex routing problems with numerous constraints, OR-Tools provides an effective framework for formulating and solving such challenges.

The tool allows the modeling of specific optimization problems through its flexible API, supporting development in Python, C++, Java, or C#. To achieve optimal performance and manage the complexity of the algorithms, the core algorithmic implementations are developed in C++. These implementations are then integrated into other development environments for practical use. This separation enables users to customize the constraints and adapt the optimization process to the unique requirements of their specific problems, offering both versatility and computational efficiency.

The solver within OR-Tools comprises three main parts:

1. **Construction Heuristic**: A method for generating an initial feasible solution to the problem.
2. **Metaheuristics**: Techniques, such as local search algorithms, that iteratively improve upon the first solution to achieve better outcomes.
3. **Constraint Programming Engine**: A framework designed to explore the solution space systematically, aiming to achieve optimality by satisfying all defined constraints.

The continuous development of OR-Tools, with frequent updates and new releases approximately every quarter, ensures its adaptability and relevance in addressing evolving optimization challenges. For this research, version 9.11 of OR-Tools was utilized, reflecting the latest advancements and features available at the time of the study.

4.1 Optimization Algorithms

Constructive heuristics for CVRP are described in a Section 4.1.1, while Section 4.1.2 focuses on the improvement heuristics applied to the CVRP.

4.1.1 Construction Heuristics

This chapter provides an overview of the constructive heuristics available in Google OR-Tools.

4.1.1.1 Greedy Heuristics

Greedy heuristic refers to an algorithmic approach that makes locally optimal choices at each step (Aron & Abraham, 2022) and it is often considered as the simplest way to solve an optimization problem. The solution of each stage is used as an input for the next stage and to find the globally optimal solution. Furthermore, a Greedy algorithm doesn't typically refine its solution based on new information. It is the heuristic that is frequently used in VRPs. The point of greedy algorithms is to prioritize local decisions to minimize current cost. In greedy heuristic, appropriate choices are selected in each step of the process in order to generate feasible solutions. To solve optimization problems, elements need to be added to the partial solution which help provide the highest gain.

Greedy Heuristics which are used in OR-Tools are described as follows:

4.1.1.1.1 Path Cheapest Arc

In the context of VRPs, Path Cheapest Arc can be compared to Nearest Neighbor heuristic. It starts at a given node (e.g., the depot) and iteratively connects the current node to the nearest unvisited node (the one with the smallest cost/distance). Besides Path Cheapest Arc, OR-Tools also provide similar Global and Local Cheapest Arcs, where in Global Cheapest Arc evaluates all possible arcs in the graph at every step and selects the one with the smallest cost, while Local Cheapest Arc program evaluates only the arcs connected to the current node at each step and selects the cheapest one.

4.1.1.1.2 Parallel Cheapest Insertion & Local Cheapest Insertion

In cheapest insertion heuristic methods, the solution is built by inserting the cheapest node at its cheapest position. The cost of insertion is based on the arc cost function. The difference between parallel and local is in the node selection for the insertion, since in the local insertion nodes are considered in their order of creation (Google OR-Tools (2024)).

4.1.1.1.3 Path Most Constrained Arc

Path most constrained arc method is similar to nearest neighbor or path cheapest arc, however not the nearest one, but the one which is most constrained node is selected, based on the constraints defined in the problem formulation.

4.1.1.2 Savings Algorithm

Savings algorithm was first proposed by Clarke and Wright in 1964, and it remains one of the most widely applied heuristics for solving VRP thanks to its simplicity of implementation and efficiency. The general idea behind the savings algorithm is to maximize the savings, either in terms of distance traveled or in the time when joining two existing routes or points, in order to decrease the collection or delivery costs (Carvalho, 2004). By minimizing the total distance traveled, the number of vehicles needed to meet all of the nodes is indirectly reduced. Therefore, the savings algorithm aims to find the route in which the accumulation of the saved distance can be maximized (Pamosoaji, Dewa, and Krisnanta, 2019).

4.1.1.3 Sweep Algorithm

Since its introduction in 1971, the Sweep Algorithm has remained one of the most widely used and efficient methods for solving various mid to large sized Vehicle Routing Problems (VRPs). Its popularity rises from its ability to generate effective routing solutions while maintaining computational simplicity. The core idea of the algorithm is to minimize the total travel distance by organizing routes based on the polar coordinate angles between each node and the depot.

The Sweep Algorithm operates in two main steps: the forward and backward procedures. In the forward procedure, nodes are clustered into routes sequentially. The process begins with the node that has the smallest polar coordinate angle relative to the depot. Nodes are progressively added to the current route as long as all constraints, such as vehicle capacity, are satisfied. Once a constraint is met (e.g., vehicle capacity is reached), a new route is initiated. This process continues until all nodes are considered and assigned to routes. Then each cluster is solved as a classic TSP problem and the final solution is obtained by summing up the total distances of all routes created.

The second step involves further refining the solution by replacing, adding, and deleting nodes within the routes, again using the polar-coordinate angle as the basis for adjustments. This refinement aims to improve the efficiency of the routes and further minimize the total distance travelled.

The backward procedure mirrors the forward methodology but operates in reverse order. Instead of starting with the smallest angle, the algorithm begins with the node having the largest polar-coordinate angle relative to the depot and proceeds in the opposite direction. The backward approach can provide alternative solutions that may outperform the forward procedure under certain conditions.

By leveraging the polar coordinate angles and following this systematic two-step approach, the Sweep Algorithm efficiently clusters nodes into optimized routes. According to Gillett and Miller (1974), its simplicity, combined with its adaptability to various constraints, makes it a reliable method for addressing the complexity of large-scale vehicle routing problems.

4.1.1.4 Christofides Algorithm

In addition to the Savings and the Sweep Algorithm, which are one of the pioneer heuristics developed for solving complex Vehicle Routing Problems, the Christofides Algorithm, introduced by Nicos Christofides in 1976, stands out for its effectiveness in solving Traveling Salesman Problems (TSP). Widely known as the 3/2-approximation algorithm, it provides a solution whose total cost is no more than 1.5 times the cost of the optimal TSP solution (Christofides, 2022).

It starts with the input graph with vertices and edge weights. Then, a Minimum Spanning Tree (MST) is constructed, which connects all vertices with the minimum total edge weight, ensuring there are no cycles. Once MSP is generated, there is no guarantee of visiting all edges exactly once. Because of it, all vertices with an odd degree, meaning vertices where the number of edges incident to them is odd, are detected.

Next step is to perform a perfect matching on the odd degree vertices of the MST. A perfect matching is a set of edges such that every odd degree vertex is connected to exactly one other odd degree vertex. The goal is to select the Minimum Maximal Matching such that the total cost (sum of edge weights) of the edges in the MMM is minimized. Furthermore, combining the edge sets of the MST and the perfect matching MMM is done, in order to form a new edge set. This edge set has two critical properties:

- It is connected (since MMM is a spanning tree and adding edges cannot disconnect it).
- All vertices now have an even degree (the perfect matching added one edge to each odd-degree vertex, making their degrees even).

A graph where all vertices have even degrees is known as an Eulerian graph. In the Eulerian subgraph, there exists an Eulerian tour: a traversal that starts at any vertex, visits each edge exactly once, and returns to the starting vertex. This property follows from Euler's theorem, which states that a connected graph with all even-degree vertices has an Eulerian cycle.

The Eulerian tour visits some vertices more than once because it follows edges repeatedly. To transform this into a TSP tour (visiting each vertex exactly once): A

shortcutting technique is applied: Skip over previously visited vertices during the traversal while maintaining the order of vertices in the tour. Since the graph satisfies the triangle inequality, the cost of the shortcut tour will not exceed the cost of the Eulerian tour.

Since the OR-Tools library is created for vehicle routing problems, a slightly different version of the algorithm is used, where any matching is possible.

4.1.2 Metaheuristics

Initial solution generation heuristics for vehicle routing problems (VRPs) typically start from an initial feasible solution and seek improvements by examining the local neighborhood of that solution. A neighborhood consists of all possible solutions obtained by applying minor adjustments or moves to the current solution, such as swapping or relocating customers between routes. If a neighborhood solution leads to an improved objective function value (e.g., reduced total distance or cost), it replaces the current solution, and the search continues from this new solution. Otherwise, the neighborhood solution is dropped, and the search continues by exploring other potential moves.

Local search algorithms, including those implemented in Google's OR-Tools, usually evaluate only a subset of the neighborhood, specifically solutions that show immediate improvement in the objective function. This selective exploration can restrict the search to a limited region of the solution space and because of this, the search might get stuck in a local minimum, meaning no immediate nearby solution is better, even though a superior solution might exist somewhere else.

To overcome this limitation, metaheuristics are used. Metaheuristics are higher-level strategies designed for complex optimization problems like VRPs, aiming to escape local minimum by temporarily accepting solutions with worse objective function values. This acceptance allows the search process to explore a broader area of the solution space, increasing the likelihood of discovering global or near global optimal solutions.

However, metaheuristics typically require specifying computational resource constraints, such as a maximum time limit. If this limit is set too short, the search might terminate prematurely, producing suboptimal results. Conversely, an excessively long

time limit might result in unnecessary computational effort without substantial improvement. Thus, selecting an appropriate time constraint is essential to balance solution quality and computational efficiency.

In summary, local search methods for VRPs effectively improve solutions but risk becoming stuck in local minimum. Metaheuristic algorithms overcome this by strategically exploring broader regions of the solution space, significantly enhancing their effectiveness in solving complex vehicle routing problems. The thoroughness of the search is often improved by allowing some moves that worsen the objective function value (Van Breedam, 2001).

4.1.2.1 Simulated Annealing

Simulated Annealing (SA) is an optimization algorithm that was developed by Kirkpatrick et al. in the 1980s to address complex search problems, particularly when the solution space is large and finding the global optimum is challenging. The algorithm is inspired by the cooling process of metals in thermodynamics, where a metal is heated to a high temperature and then cooled slowly. During this process, the atoms in the metal move freely at higher temperatures, occupying higher-energy, less stable positions. As the temperature decreases, the atoms gradually settle into a low-energy, stable configuration, such as a perfectly solidified crystal. If the cooling occurs too quickly, the atoms can become trapped in higher-energy states, resulting in a less stable structure.

Simulated Annealing mimics this process for optimization by modeling the exploration of possible solutions as a series of transitions between states with different “energy” levels, where energy corresponds to the cost of a solution. The Metropolis algorithm underpins this behavior: if the system is in one state with a certain energy and transitions to another state with a different energy, the decision to accept the new state depends on the energy difference. If the new state's energy is lower (indicating a better solution), it is always accepted. Conversely, if the energy is higher (indicating a worse solution), it may still be accepted, but with a probability that decreases as the energy difference increases. This probability is also higher at higher temperatures, allowing the system to explore a broader range of states early in the process. As the temperature decreases, transitions to worse solutions become less likely, encouraging the system to stabilize.

In the context of optimization problems, the goal of Simulated Annealing is to find a solution with the lowest cost or the highest quality, depending on the objective. The algorithm starts with an initial solution, an initial temperature, and a set of stopping criteria. At each temperature level, the algorithm generates a new solution by making a slight modification (perturbation) to the current solution. The quality of the new solution is compared to the current one: if the new solution is better, it is always accepted. If it is worse, it may still be accepted with a probability determined by the temperature and the degree of deterioration in the solution. This probabilistic acceptance of worse solutions helps the algorithm avoid becoming trapped in local minima, where the solution is optimal only within a small neighborhood of the search space.

The process continues iteratively, with the temperature gradually decreasing after a fixed number of iterations at each level. As the temperature decreases, the algorithm explores the solution space less freely, focusing more on refining the current solution and making incremental improvements. The algorithm stops when one of the predefined stopping criteria is met, such as reaching a very low temperature, completing a maximum number of iterations, exceeding a time limit, or observing no further improvement in the solution.

Simulated Annealing balances exploration (searching broadly across the solution space) and exploitation (focusing on refining good solutions) through its temperature-based mechanism. This balance makes it an effective technique for solving complex optimization problems, enabling it to escape local minima and converge toward an optimal or near-optimal solution.

4.1.2.2 Tabu Search

Tabu Search (TS), introduced by Glover in 1986, is a heuristic optimization technique designed to overcome the challenge of getting trapped in local optima. It has proven highly effective for solving large-scale combinatorial problems. The core idea of Tabu Search is to extend the search process beyond local optima by allowing moves that do not immediately improve the solution. To prevent the algorithm from retracing its steps and revisiting previously explored solutions, certain moves are marked as “tabu” (prohibited). These tabu moves typically reverse the effect of recent transformations

and are restricted for a limited number of iterations, a duration known as the “tabu tenure”.

Tabus serve a role in steering away the search from previously visited areas of the solution space, thereby enabling a broader exploration. The algorithm relies on a short-term memory structure, commonly referred to as the ”tabu list”, to store these restrictions. This memory is finite, recording only a small portion of recent changes to the solution, which helps to manage computational efficiency while still effectively avoiding cycling back to prior solutions. The most frequently used tabus involve recording recent transformations and prohibiting their exact reversal.

While tabu restrictions are central to the functioning of Tabu Search, they can sometimes be overly rigid. In some cases, they might block promising moves that would otherwise lead to significant improvements or cause the search to stagnate. To counteract this limitation, the algorithm uses aspiration criteria, which provide a mechanism to override tabu restrictions under certain conditions. The most common aspiration criterion allows a tabu move if it results in a solution with a better objective value than the best-known solution. This ensures that the algorithm remains flexible and can adapt when a tabu move offers the potential for significant improvement. The guiding principle is that if there is no risk of cycling, tabu restrictions can be disregarded.

In summary, Tabu Search effectively balances exploration and restriction, using tabu lists to avoid revisiting previous solutions while employing aspiration criteria to maintain adaptability. This combination allows Tabu Search to navigate complex solution spaces efficiently and discover high-quality solutions.

4.1.2.3 Guided Local Search

Similar to the above-mentioned algorithms, Guided Local Search (GLS) stands out as one of the most effective metaheuristics for avoiding local optima in complex vehicle routing problems. GLS is derived from GENET, an approach developed by Tsang and Wang (1991) for solving optimization problems with a finite set of constraints. A reason for the popularity of GENET lies in its learning rule concept and its ability to adapt and learn from mistakes by modifying penalties for rule violations. This approach allows it

to overcome dead ends and eventually find a solution that satisfies all constraints in a problem.

Voudouris and Tsang (1998) stated that the GLS approach iteratively explores and refines solutions by introducing penalties as a guiding mechanism. The basis of the algorithms is the employment of an augmented cost function, which enhances the original objective function by incorporating penalties for violating specific rules. These penalties serve to redirect the search away from less desirable solutions and encourage exploration of new areas within the solution space. The dynamic adjustment of these penalties is one of the main advantages of this algorithm. Each node within the problem is assigned a cost that shows its impact on the overall solution cost, leading the algorithm to prioritize the nodes most critical to penalize during the search.

Initially, all penalties are set to zero, ensuring the augmented cost function is equal to the original objective function. When the search becomes trapped in a local optima, penalties are increased for the nodes contributing to that solution. This adjustment effectively discourages the search from revisiting the same local optima. The process repeats iteratively until either a satisfactory solution is found or a predefined stopping condition is reached.

Even though GLS, Tabu Search and Simulated annealing share the same main purpose to avoid getting stuck in local optima and finding near optimal solutions efficiently through the iterative search strategies, they differ significantly in how they navigate the search space. In GLS, local optima is avoided by introducing penalties to undesirable solution nodes and the approach is activated when local search settles in a local optima and the penalties guide the search toward exploring new areas of the solution space until it escapes. On the other hand, in Tabu Search, a tabu list is used to forbid revisiting recently explored solutions or moves, effectively guiding the search toward new areas, and its approach is usually activated in every iteration and unlike in GLS, ensuring that the search does not settle in a local optima. Simulated Annealing (SA) takes yet another approach, by using randomness to escape local optima. This randomness allows us to accept worse solutions with a certain probability, which decreases as the “temperature” cools over time. This probabilistic acceptance supports broad exploration of the search spaces early in the process, helping the algorithm move away from local optima earlier.

5. OpenRouteService

In modern logistics and operations research problems, access to real-time geographical data is essential for solving real-world challenges like determining optimal routes and minimizing travel time or distance. In this project, OpenRouteService plays a crucial role in creating driving mode distance matrix and providing detailed mapping services with features such as directions for different modes of transport, detailed information board and intuitive, route-selective visualization. Its accessibility as a free source and reliability make it a great choice for integrating mapping functionalities into our problem.

ORS is a platform which provides different API features using Open Street Map Data which can further be used in different processes and scenarios. These various services such as Directions, Matrix or Geocode Services are developed by HeiGIT, Heidelberg Institute for Geoinformation Technology (Heidelberg Institute for Geoinformation Technology [HeiGIT], n.d.).

The OpenRouteService Directions service provides visualization of routes and navigation details, offering diverse travel options suitable for multiple modes of transportation. Users can select from cars, various bicycle and pedestrian categories, wheelchairs, and heavy vehicles. Each travel mode utilizes a specifically tailored street network designed to meet its setting requirements. To further personalize routes, the API allows users to customize parameters such as road type restrictions, vehicle attributes, summary information, legends, and more.

Another used service from OpenRouteService is the Matrix API, which calculates distances between a set of defined locations (origins and destinations) and returns the results in a structured JSON format. This API is highly convenient and scalable, especially designed for batch requests with the aim to compute aggregated route metrics. A key advantage of this service is the ability to specify the transportation mode, ensuring the calculated distances reflect the chosen mode accurately.

Typically used for generating distance matrices in problems like the Traveling Salesman Problem (TSP), this service supports up to 2,500 distance pairs per request, allowing for combinations of up to 50 origins and 50 destinations simultaneously. By

default, a square duration matrix is returned, pairing each location with every other. If a distance or duration cannot be determined, the result will appear as null.

In our specific use case, since the number of requested origin-destination pairs significantly exceeds the limit of a single request, the ORS Matrix API divides the total matrix into multiple smaller batches. Each batch independently computes distances or durations for its subset of pairs. Once all batches are processed, ORS assembles these partial results into a unified distance matrix. This approach guarantees that each origin-destination combination is represented precisely once, resulting in a complete and accurate final matrix without overlaps or omissions.

The process begins with obtaining an API key from the OpenRouteService, which is required to access its services like the Matrix API and Directions API. To integrate OpenRouteService features with Python, input data is prepared in the form of a file containing the coordinates of points (e.g., containers and landfills). This file can be created manually using tools like Excel or generated programmatically within Python for larger datasets or dynamic scenarios.

Once the input data is prepared, it is processed in Python. Libraries like *pandas* are used to read and manipulate the data, while Google's OR-Tools optimization algorithms are applied to solve the routing problem. Python handles the data efficiently, generating solution routes while considering various constraints.

To calculate distances between points, Python communicates with the OpenRouteService API using libraries such as *openrouteservice* and *requests*. The Matrix API plays a crucial role here, providing precise travel distances between multiple points. By combining this data with optimization algorithms, the most efficient routes are determined. It is important to mention that real-time traffic data with this API could not be retrieved.

The final step involves visualizing the solution. Python creates output files containing the optimized routes and corresponding coordinates in formats like *JSON, CSV,* or *GeoJSON*. These results can then be visualized using libraries such as *folium* where visualization features are included, which render the routes directly on a map. By marking locations such as containers and landfills on the map alongside optimized routes, the service transforms complex VRP challenges into user-friendly and easily

understandable solutions. This not only adds value to the overall research but also makes these complex problems more accessible to others.

6. Solution Framework for Solving Periodic Vehicle Routing Problem

In this section the solution approach that was used to solve the problem is explained and described. Having introduced the problem itself in Chapter 1 and reviewing literature which provides relevant research that suits our Vehicle Routing Problem, we set the foundation for this study. Furthermore, a detailed problem description is given in Chapter 3, while in Chapters 4 and 5 an introduction to Google OR-Tools and OpenRouteService platform is provided.

Considering that PVRP is an extension of classic VRP and the complexity given by the size of real-life problems, we applied a decomposition, two-stage method, where before solving vehicle routing problems, feasible visiting schedules must be created in the customers assignment phase. In the first phase, customers are assigned to days based on the given frequency over the planning horizon, while in the routing phase, routes are constructed considering generated day schedules.

The approach implemented consists of two main components, which are described as follows: Firstly, an initial solution is generated. Subsequently, the components of the Large Neighborhood Search scheduling algorithm and scheduling process are examined and lastly, components of Google OR-Tools solver and routing process are presented.

The two-stage approach (scheduling and routing) is described in a multi-trip VRP setting. For the benchmark instances, we apply the simplified framework to the simpler single-trip version of the problem, whereas for the real-life case study, the full multi-trip functionality is implemented as described.

6.1.1 Scheduling Phase

- Initial Solution

The scheduling phase begins with the construction of the initial solution which serves as a starting point for the implementation of the Large Neighborhood Search Algorithm and as such it must respect the constraint that daily vehicle capacities must be respected. To generate the initial solution a greedy heuristic was used to assign

containers to visiting schedules by respecting their frequency and available visiting patterns. Firstly, customers are sorted in a descending order based on their volume, meaning containers with highest volumes are prioritized over others, as they are less flexible and fit the schedule harder. For each customer in the defined order, the algorithm checks all possible visiting patterns and calculates the distance costs for each available pattern, selecting only patterns which do not violate the daily capacity constraint. In each iteration, the algorithm chooses the cheapest feasible pattern and updates the daily scheduling assignment. If a container cannot be assigned because there is not enough capacity, the algorithm tries a redistribution. In this phase, it searches for the already assigned containers which have the most flexible patterns, thus freeing up spaces for the unassigned ones while searching cheapest distances.

Once the initial visiting schedule is generated, it is further improved using the Large Neighborhood Search Algorithm. Originally proposed by Shaw (1998) and later enhanced by Ropke and Pisinger (2006), LNS is an iterative metaheuristic designed to improve an incumbent solution by exploring large portions of solution space. In each iteration, the algorithm applies two phases, a destroy and a repair operator. In the destruction phase, a part of the current solution is removed based on the defined removal operator, leaving a set of nodes unassigned. In the subsequent, repair phase, these unassigned nodes are reinserted into partial solution using a repair operator. Through this destroy-repair mechanism, the algorithm is able to escape a local minima and explore other neighborhoods. A new solution is accepted, if it meets a predefined acceptance criteria.

For the purpose of this research, a version of LNS with several enhancements is implemented. The solution approach is depicted in Figure 3.

Algorithm 1 Scheduling Phase LNS

Input: C customers, D days, distance matrix $Dist$, vehicle capacities cap_d, number of vehicles m
Output: best pattern assignment π^*, day assignments S^*
for each $d \in D$ **do**
    $\hat{cap}_d \leftarrow m \times cap_d$
end for

Generate initial solution:
for each $c \in C$ *(sorted by demand)* **do**
    for each $p \in P_c$ **do**
        if $feasible(p, \hat{cap})$ **and** $|days(c,p)| = freq_c$ **then**
            assign p to c
            update *day_assignments* and day loads
        end if
    end for
end for
while *time limit not reached* **do**
    Use Destroy Operator: remove $\alpha\%$ of assigned customers
    Use Repair Operator: reinsert using regret insertion (respecting capacity & frequency)
    $u_{cur} \leftarrow \#unassignedincurrent, \quad u_{new} \leftarrow \#unassignedinnew$
    $dist_{cur} \leftarrow$ depot–return distance approx. (per day: depot legs + MST + min return)
    $dist_{new} \leftarrow$ same on new assignments
    $\Delta \leftarrow dist_{new} - dist_{cur} - \beta\,(u_{cur} - u_{new})$
    if $u_{new} < u_{cur}$ **or** $\Delta < 0$ **or** accept by probability **then**
        accept new solution
        if best so far **then**
            update best solution
        end if
    end if
    update temperature $T \leftarrow T \cdot cooling\_rate$
end while
return best pattern assignment, day assignments
▷ Day assignments S* passed to OR-Tools VRP solver for routing optimization

Figure 3 Scheduling Phase Pseudo Code

As already described, the greedy initial solution is constructed, whereas containers are sorted based on volume and frequency flexibility. In each iteration, every visiting pattern is checked against daily capacity constraints and day assignments are updated accordingly.

- Objective Function

The objective function is to create feasible daily schedules where the total route length is minimized.

- Destroy Operator

In this framework, a simple destroy operator, *random removal* is used. It selects 30% of the neighborhoods randomly on each day of the planning horizon and removes them from the solution. In this way, a solution is diversified, enabling intensive exploration of a search space and provides scalability, avoiding performance issues regarding problem size.

- Repair Operator

The repair operator uses *regret-based insertion heuristic,* where regret heuristic evaluates for each removed container regret value. A regret value represents insertion cost differences between the cost of inserting container *i* in its best position, and cost of inserting the container *i* in its second best position. In each iteration the container *i* that has the maximum regret, meaning highest costs of not inserting it immediately in the route is chosen.

$$\Delta \leftarrow dist_{\text{new}} - dist_{\text{cur}} - \beta\,(u_{\text{cur}} - u_{\text{new}})$$

The process starts with the calculation of regret values for all unassigned containers. Containers are sorted in a descending order, where ones whose exclusion would lead to the highest routing costs are prioritized. Furthermore, each container is inserted into the solution considering the most cost-effective feasible visiting pattern. The process is repeated until no more containers can be inserted.

- Acceptance Criteria

As for the acceptance criteria, a three level procedure with greedy and probabilistic approaches is used. As a highest priority, a solution with fewer containers assigned is always chosen, regardless of distance costs. Furthermore, if the number of assigned containers is the same as in the previous iteration, but the distance is lower, it is also accepted. However, if the new solution is worse, a simulated annealing approach is used to try to optimize the distances.

The acceptance criteria is set to accept the new non-improvement solution *x`* with the probability of

$$e^{-\frac{f(x')-f(x)}{T}},$$

where T represents temperature and it must be higher than zero.

Once the acceptance probability is calculated by comparing the current and a new solution cost, it is compared with the random threshold generated between 0 and 1, where if the acceptance probability is higher than the threshold, the new solution is accepted, otherwise the search continues with the current solution.

The initial temperature starts at 1000, and decreases by 0,1% with each iteration. The temperature decreases based on the formula $T=T \times c$, where cooling rate represents c, and it is updated with each iteration, regardless whether the container is accepted or not. The procedure continues iteratively until the predefined time limit is reached, with the temperature gradually decreasing, transitioning from exploration to exploitation approach.

- Stopping Criteria

The stopping criteria is reached when the time limit has passed.

The scheduling phase outputs daily container assignments as *day_assignments* containing container indices assigned to each day in the horizon. These results are immediately considered by the routing phase through a direct data transformation in the *create_multitrip_data_model_ortools()* function, where index 0 represents the depot and container indices are adjusted for the distance matrix format. This transformed data structure serves as input to the OR-Tools RoutingModel, which solves the VRP for each day independently using the container assignments determined by the LNS scheduling algorithm.

6.1.2. Routing Phase

As Periodic Vehicle Routing Problem is a problem with two-stage approach, once scheduling is completed, routing phase takes place.

In this chapter, the process of solving VRP using Google OR-Tools solver is outlined from the problem initialization to the solution visualization.

Data Definition

The process starts by defining the problem data, where *create_multitrip_data_model()* function stores the input data such as:

- Distance Matrix
- Depots, Containers
- Vehicles
- Volumes
- Vehicle Capacities

- Distance Matrix Scaling

Since the routing solver works explicitly with integer values, any decimal value in the distance matrix needs to be rounded. By rounding the values, distortions are created and therefore the quality of the solution is under the question.

To mitigate it, as in other optimization approaches, the best practice is that the entire distance matrix is multiplied with a constant factor, e.g. 100. This scaling procedure does not alter the solution, as in the post-processing stage it is adjusted again. In the print method, solution routes are divided with the same scaling factor, ensuring the routes considered distances in their original value.

Callbacks

Once data is created, to make the routing solver run, **callbacks** and **dimensions** must be defined. A callback represents a function that solver calls during the optimization to evaluate costs, demands, constraints, etc. and acts like a bridge between solver and the problem setup data.
The most common used callbacks in VRP problems are:

1. Transit Callbacks
2. Unary Callbacks

Transit Callback takes any pair of nodes and calculates the distances between them. It is recognized by the solver as the *transit_callback_index* function, and it can be used anytime distances between two nodes must be calculated. The function takes two indices *from_index* and *to_index* as inputs and returns the distances from the distance

matrix.
To tell the solver how to calculate the costs of each arc, the *Arc Cost Evaluator* is used. If the cost between a pair of nodes is defined as a distance between the nodes, then the arc cost evaluator is represented as the *transit_callback_index* and a method *routing.SetArcCostEvaluatorOfAllVehicles ()* is used. However, if costs of arcs differ between vehicles and for example different vehicles have different speeds, thus time needed for vehicles is different, for each vehicle a separate arc cost evaluator is used and represented by the method *routing.SetArcCostEvaluatorOfVehicle ().*

In our problem the only transit callback implemented is the ***Distance Callback***, which was used to calculate distance costs between each pair of nodes, either distances between containers or distances between container and depot. This function is a place where the distance matrix must be translated to the solver notations (indices), where the distance matrix scaling is happening and floating numbers are converted to integers. The distances are also converted back to real-life distances in the same function, providing more accurate data in the solution.
This function is called by the solver, which then calculates the total route costs and uses this callback to sum the distances provided, as well as compare all possible routes in order to get the solution that will minimize the objective function.

On the other hand, **Unary Callbacks** take a single node index and return a value associated with visiting exactly that location. To make it recognizable for the solver, a method *routing.RegisterUnaryTransitCallback ()* is used, and it is used mostly to represent demand, service time, time window etc.
Unary Callbacks are primarily used with the **dimensions**, which are used to track cumulative values along routes.

In our problem two unary callbacks were implemented, ***Demand Callback*** and ***Container Count Callback***.
The purpose of the *Demand Callback* was to enable the solver to distinguish between depot, containers and reload (dummy) depots. It returns different values based on the node classification:

1. Containers are treated with their exact demand
2. Depot with zero value

3. Reload (Dummy) depots with negative values (-15.000, a capacity of the vehicle)

Furthermore, this function enables the proper work of capacity dimension described in the chapter below. More specifically, the demand callback creates a capacity reset mechanism by introducing negative demand values at specific nodes, allowing the reduction in the cumulative load during the route. This allows the vehicle to completely unload at the reload (dummy) depot, therefore resetting its full capacity.
The *Container Count Callback* is implemented as the foundation for the *Container Count Dimension* described below. The purpose of the callback is to identify containers, returning 1 for container visits, and 0 for depots.

Dimensions

Dimension is an object generated by the routing solver in order to track cumulative values along vehicle routes, such as total time travelled, demands collected, etc. It is a mandatory object, whenever the problem contains such quantities which has to be calculated cumulatively, either as a constraint or in the objective function. To create a dimension, method *Add.Dimension ()* is used. When defining a dimension, several parameters must be maintained.

Parameters:

1. *callback_index* - The index for the callback that stores the value.
2. *slack_max* - A variable used for tracking buffer capacity.
3. *capacity* - The parameter that represents the maximum quantity cumulated per route, whether it is vehicle capacity, customer capacity or any other constraint.
4. *fix_start_cumulative_to_zero* - This parameter controls whether the cumulative value must start at 0 capacity at the beginning of each route. It can be set either as true or false.
5. *dimension_name* - A string label used to identify the dimension within the program.

These parameters are needed to create a basic constraint system that enforces hard constraints. However, the solver provides two configuration methods that enable

tailoring of the problem constraints, thus allowing more flexibility or control. If we want to set up special rules and use penalties, a method *routing.GetDimensionOrDie* is essential, as it transforms hard constraints into soft ones, therefore increasing model flexibility and improving the possibility for obtaining more promising solutions.

Capacity Dimension

In our real-life problem, the core challenge was to extend the standard, single-trip capacitated VRP into a multi-trip framework. This was achieved by implementing dimensions that interact with callbacks in the solver. In the problem, two custom dimensions were created, of which the *capacity dimension* plays the pivotal role in enabling the multi-trip functionality.

The *capacity dimension* was registered through the solver using the *demand callback*, which, as explained, tracks cumulative vehicle load along the route. It ensures that vehicle load is updated after every container visited, and that at each point of time, no vehicle exceeds its physical capacity. While in classical single-trip VRP this would be sufficient, in the multi-trip approach additional flexibility was required. Three parameters influenced the behavior of this capacity dimension:

6. *Slack_max*
   This parameter allows the temporary fluctuations in cumulative load values during the solver`s internal iterations. In the single-trip approach, *slack_max = 0,* which forbids any negative value. For the multi-trip problem, however, *slack_max* is set equal to vehicle capacity. This adjustment is essential when vehicles encounter reload depots, which are modeled with the negative demand of -15.000 litres. When visiting such a node, the cumulative load temporarily becomes negative, showcasing the unloading of the vehicle. This configuration of the *slack_max* parameter allows these negative temporary values without the solver rejecting the solution as infeasible.
7. *Vehicle_Capacity*
   This parameter represents the hard constraint and it ensures that at no point of the route can the load exceed the maximum vehicle capacity. Even though negative cumulative values are permitted after the unloading, the solver ensures

that all subsequent loading cycles remain within the vehicle`s physical capacity.

8. *Start_Cumulative_To_Zero*
   By setting this parameter to true, it is enabled that every vehicle begins its route with zero load. Since waste collection operates in a way that every night trucks start empty and must unload at the depot before starting a new trip, this was a logical decision.

Interaction of these three parameters with different node types enables the multi-trip approach. Container volumes add positive demand, start and end depot contribute zero demand, and reload depots add negative demands, enabling the reset of vehicle capacity. This approach allows the solver to treat a sequence of trips as a single continuous route with unloading moments incorporated in it. Important to mention, each visit to a reload depot incurs real travel distance, which is accumulated through the *distance callback*. Since the objective is to minimize the total distance travelled, the solver avoids unnecessary reload depot visits unless they are forced by the vehicle capacity limitation.

Container Count Dimension

While the *capacity dimension* enables multi-trip functionality, the *container count dimension* addresses the issue of workload balancing across the fleet. When relying solely on distance minimization, the obtained solutions always assigned a disproportionately large number of containers to certain vehicles, while remaining others completely underutilized. Such imbalances do not reflect realistic situations, and therefore this dimension had to be considered in the model.

This dimension accumulates values from the *container count callback*, increasing by 1 for every container visited, while ignoring depot and reload depots values. In this way, it calculates cumulatively the number of containers visited by each vehicle across the entire planning horizon, including all trips and reload cycles. Unlike *capacity dimension*, container counting persists across reload operations, ensuring that each vehicle in each trip is considered for its total workload. The dimension is configured with the following parameters:

1. *Slack_max*
   This parameter is set to 0, as containers/depots visit is either 1 or 0.
2. *Container_Count*
   This parameter is set as equal to the total number of containers in the instance, providing an upper bound that ensures feasibility.
3. *Start_Cumulative_To_Zero*
   Setting this parameter to true, it is guaranteed that all vehicles begin their daily routes with zero containers visited, creating a stable baseline for workload balancing.

This dimension integrates with workload balancing through soft constraint mechanisms that encourage even distribution of containers across available vehicles.

Penalties

Constrained VRPs might get easily infeasible when problem parameters exceed the solver capabilities. A periodic multi-trip VRP is complex by nature, and as such, our problem has several objectives: capacity constraint must be respected, secondly, workload balancing is required and lastly, all containers must be visited, since each container must be regularly emptied. Considering that this type of the problem suppresses a traditional single-trip setup, we introduced a three-tier penalty approach to prevent the solver getting stuck and running for a long time finishing with no solution. Three penalty approaches that were used in the approach and are provided by OR-Tools are:

1. Soft Upper Bounds
2. Soft Lower Bounds
3. Disjunctions

Soft Upper Bounds

A Soft Upper Bound is a penalty which allows exceeding the limit but charges a penalty for each unit of violation. We implemented a soft upper bound penalty of 50,000 cost units for vehicles serving containers beyond the threshold maintained. The threshold which represents the number of served containers is calculated dynamically as a

maximum of 10 containers per route, or total number of containers scheduled for a day divided by 4, where the goal was to balance the workload distribution. A penalty of 50,000 cost units per vehicle usually is classified as high penalty value and it ensures that the workload balancing is violated only when mathematical constraints make balancing impossible and then primary objective is to serve all the customers, regardless of the fair distribution.

OR-Tools Methods Used:

- *SetCumulVarSoftUpperBound()*: Applies penalty for exceeding container threshold

Soft Lower Bounds

Soft Lower Bounds penalizes vehicles staying below a minimum threshold, encouraging the solver higher utilization. A penalty of 10,000 cost units addresses the vehicles underutilization by penalizing those vehicles which do not serve any container in the iteration/solution. This penalty has a lower priority when compared to the upper bound penalty, and shows that overutilization prevention takes precedence over underutilization avoidance. Maintenance of this penalty was critical, keeping in mind that only 7 vehicles are available and that routing happens only during the night, so it must be completed in a certain time horizon. It discouraged the solutions where some vehicles remain completely unused while others become overloaded.

OR-Tools Methods Used:

- *SetCumulVarSoftLowerBound()*

Disjunctions

Disjunctions represent variables that the solver uses to make decisions which locations to include in routes. For each location a decision variable is introduced, and it determines whether the location is visited or skipped. Once these penalties are introduced, the objective function is calculated considering not only costs between locations, but also penalty costs for all skipped locations. These penalties are added using the *AddDisjunction* method. In our problem two disjunction penalties are implemented:

Dropping Containers Penalty

The dropping containers penalty of 100,000 costs is a highest priority penalty and it represents the critical importance of including all containers to the routes, exceeding the sum of all other costs related to problem setup, ensuring that the exclusion of the containers happens only under extreme unfeasible situations. Setting up this penalty helped the solver to treat the problem as multi-trip with reload depots, where vehicles, once capacity is reached, can return to a depot to reset their load. These dummy depots are introduced at the same locations of the original depot and they can be skipped in a route at the cost of 0. In situations when reload depots cannot provide sufficient route capacity to visit all containers scheduled for a day, this high penalty forces the solver to explore all possible combinations, before dropping some containers from the solution.

Reload Depot Penalty

The purpose of this penalty is to enable a route capacity reset when beneficial for total route optimization. If the solver decides to drop the reload depot at any point, it gets assigned 0 cost. This decision is completely optional, without any penalty implication.

OR-Tools Methods Used:

- *routing.AddDisjunction()*: Makes nodes optional with specified penalty costs

Search Limits

In the solver two approaches to limit the search duration exist:

1. Time Limit (*search_parameters.time_limit.seconds*)

This parameter is used as a stopping criteria in our problem, and it terminates solution search and returns the best solution found within the allocated timeframe. In our approach, different timeframes were used in order to track the solution improvements. Thus routing time limits starting from 30 seconds per day until 1800 seconds were used and evaluated.

2. Solution Limit (*search_parameters.solution_limit*)

The solution limit, on the other hand, controls the number of feasible solutions generated before termination.

Solution Search Strategies

As above-mentioned, OR-Tools support different heuristics and metaheuristics for generating initial and improvement solutions. Initial solution strategy in the solver is defined as a first solution strategy, where in our problem, a parameter *search_parameters.first_solution_strategy* is set to Path Cheapest Arc heuristic. For the improvement solution and local search Guided Local Search metaheuristic is used.

OR-Tools Implementation

The routing phase constructed in this research represents an enhancement of the standard OR-Tools VRP approach in order to consider specific multi-trip requirements. The implementation begins with the definition of the input data and the transformation of the multi-trip logic into the single-trip approach that is solver compatible. To achieve this, a four-tier node had to be introduced. Start and end depot, meaning each route starts and ends at the depot, containers and dummy, reload depots. The distance matrix is then adjusted, with infinite costs assigned to the routes which are not operationally feasible. In this way, the solver is forced to make realistic routing decisions and to add reload depots when capacity limit would be violated otherwise. After the problem definition, the *RoutingIndexManager* and *RoutingModel* are created in order to ensure correct mapping between problem nodes and solver indices. At this stage, distances between nodes and container volumes are provided to the solver through defined callbacks, allowing the RoutingModel to operate with integer-scaled data. By calling the function *SetArcCostEvaluatorOfAllVehicles(transit_callback_index)*, OR-Tools automatically sets the objective to minimize the total travel distance. The next step includes the introduction of constraints as dimensions. Alongside the capacity and container count dimensions, which ensure the vehicle capacities are not violated and workloads are balanced, penalties are introduced as an additional mechanism by discouraging workload imbalances. To avoid infeasible solutions, disjunction is also added, assigning high penalties to routes that excluded containers. Once the problem is defined, search strategies are applied. The solver firstly generates an initial solution with the Path Cheapest Arc construction heuristic. This solution is then improved by the Guided Local Search metaheuristic. The final steps involve solution parsing and reconstruction of multi-trip routes.

Algorithm 3 OR-Tools Multi-Trip VRP

Input: base distance matrix D, demands q, vehicle capacity Q, vehicles m, depot, day_indices, time_limit

Four-tier nodes: starts, containers, reloads, ends
Custom matrix: set ∞ for invalid arcs
manager $\leftarrow$ RoutingIndexManager; routing $\leftarrow$ RoutingModel
register distance & demand callbacks; set arc costs
add Capacity & ContainerCount dimensions (soft bounds)
add disjunctions (reloads: 0, unvisited containers: high penalty)
set search (PATH_CHEAPEST_ARC + GUIDED_LOCAL_SEARCH, time_limit)
solution $\leftarrow$ SolveWithParameters
if solution = None **then**
    return None
end if
for each vehicle **do**
    parse route into trips using reload/end as boundaries
end for
print totals and return trips

Figure 4 Pseudo Code for solving the Multi-Trip Routing Problem with OR-Tools

7. Computational Results

The code utilized in this research has been implemented in Python 3.10. All algorithms were executed on a PC equipped with an Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 2101 MHz, with four physical cores, eight logical processors, and 16 GB of RAM.

While the overall methodology is based on multi-trip approach, it is important to note that benchmark instances are solved in their single-trip form to allow comparability with existing studies. The real-life case study results apply the complete multi-trip framework.

7.1 Benchmark Instances

To evaluate the performance of the Google OR-Tools solver, instances introduced by Cordeau et al. (1997) were used. Out of the 40 available instances, 15 were selected to test the solver's capability in the PVRP. These instances were originally created for Multi Depot and Periodic VRP under the single-trip assumption (approach). Accordingly, input data and the solver setup were designed to handle the instances of a single-trip PVRP.

As shown on Table 1, each instance in the input data file begins with the header row that defines the basic configuration of the problem (HEC Montréal, Chaire Distributique n.d.). There are four parameters maintained, the type of the VRP problem *type*, the number of vehicles *m*, the number of customers *n* and the number of days *t* in the planning horizon. The *t* subsequent rows specify daily constraints, such as the vehicle capacity *Q*, and if applicable, the maximum route duration D, which is set to value greater than zero in the problems containing time constraints. Each customer in the instance includes a unique ID *i*, *x* and *y* coordinates, service duration *d* (if applicable), demand *d*, the frequency of visits *f*, the number of possible visiting combinations *d* and the *list*, which provides the list of these combinations. To represent the visiting combinations across the planning horizon a binary coding system is used. A value of 1 indicates the customer is visited on that day, otherwise it is 0. For instance, if a customer is to be visited on days 2 and 4 in a 5 day planning horizon, the visit pattern is 01010 (HEC Montréal, Chaire Distributique n.d.). In the data files, since using binary

strings could be complex, the instances instead use the decimal equivalent of each string.

Header Row			
Type (type)	**Number of Vehicles (m)**	**Number of Customers (n)**	**Number of Days (t)**
1	3	50	5
Duration and Capacity Constraints			
Maximum Duration of Route per Day (D)		**Maximum Vehicle Capacity per day (Q)**	
0		160	

Customer Information							
Customer ID (i)	**X Coordinate (x)**	**Y Coordinate (y)**	**Service Duration (d)**	**Demand (d)**	**Frequency of Visit (f)**	**Number of Combinations**	**List of Possible Visit Combinations (in decimal)**
0	30	40	0	0	0		
1	37	52	0	7	1	5	1, 2, 4, 8, 16
2	49	49	0	30	5	1	31

Table 1 Input Data File Structure

Table 2 provides an overview of the characteristics of the selected instances.

I	*n*	*m*	*t*	*Q*
p02	50	3	5	160
p03	50	1	5	160
p04	75	6	5	140
p07	100	4	2	200
p09	100	1	8	200
p11	138	4	5	235
p14	20	2	4	20
p15	38	2	4	20
p16	56	2	4	40
p17	40	4	4	20
p18	76	4	4	30
p19	112	4	4	40
p20	114	4	4	30
p22	114	6	4	30
p23	168	6	4	40

Table 2 Selected Instances Characteristics

7.1.1 Results

For validation purposes, results obtained from the solver are compared with the best known solution from the literature. Table 3 presents an overview of the results from PVRP instances provided by Cordeau (1997). Regarding the origin of the instances, instances 2-9 originate from Elion et al. (1974), while instances from 14 to 23 were drawn from Chao et al. (1995).

For each instance, the total route length from the best known solutions is shown in the column BKS, while the results from the LNS scheduling and Google-OR Tools routing algorithm are listed in column LNS + Google-OR Tools.

It is important to note that all results displayed were generated in solver runs with the computational time limits of 720 seconds for scheduling and 60 seconds for routing per instance. Consequently, they may not represent the absolute best-known solutions for each instance. For the generation of the initial routing solution Google-OR Tools used *Path Cheapest Arc* algorithm across all instances. Different improvement strategies were subsequently applied and the best-performing algorithms varied across instances. Although different strategies were used, in 7 out of 15 instances *Tabu Search* and *Guided Local Search* reached the same result, reaching the best known solution for instances 15 and 16. Tabu Search achieved the best results for instances 3,7,22,23; while for instances 4 and 11, Guided Local Search constructed the best solution within defined search time limits.

To assess the performance of the solver, the percentage deviation in the total route length from the best solutions found was analyzed. As shown in Table 3, the overall performance can be considered as very good, as in all instances the deviation from the best-known solutions was less than 10%. For 8 instances the deviation is lower than 5%, where 2 instances yielded the best known solutions from the literature. These results prove that the different metaheuristics available are capable of generating high-quality solutions within reasonable computational time.

Although the results obtained are highly promising, there is still potential for further improvements. With the extended solution search limits and different combinations of initial and improvement routing strategies it is realistic to expect the results which are even closer to the best-known solutions.

I	*n*	*m*	*t*	*Q*	BKS	LNS + Google OR-Tools	Delta	*Dev (%)*
p02	50	3	5	**160**	**1,322.9**	1,405.5	*82.7*	*6.25%*
p03	50	1	5	**160**	**524.6**	566.5	*41.9*	*7.99%*
p04	75	6	5	**140**	**835.3**	856.9	*21.6*	*2.59%*
p07	100	4	2	**200**	**826.1**	852.3	*26.2*	*3.17%*
p09	100	1	8	**200**	**826.1**	870.3	*44.2*	*5.35%*
p11	138	4	5	**235**	**775.8**	853.0	*77.2*	*9.95%*
p14	20	2	4	**20**	**954.8**	974.1	*19.3*	*2.02%*
p15	38	2	4	**20**	**1,862.6**	**1,862.6**	***0.0***	***0.00%***
p16	56	2	4	**40**	**2,875.2**	**2,875.2**	***0.0***	***0.00%***
p17	40	4	4	**20**	**1,597.7**	1,733.6	*135.9*	*8.51%*
p18	76	4	4	**30**	**3,131.1**	3,215.4	*84.3*	*2.69%*
p19	112	4	4	**40**	**4,834.3**	4,905.6	*71.1*	*1.47%*
p20	114	4	4	**30**	**8,367.4**	8,412.0	*44.6*	*0.53%*
p22	114	6	4	**30**	**4,193.9**	4,559.8	*365.6*	*8.72%*
p23	168	6	4	**40**	**6,420.7**	7,000.0	*565.9*	*8.80%*

Table 3 Comparison with Best Known Solutions

7.2 Real-Life Instance

As outlined in the problem description, Novo Sarajevo, district of Sarajevo Canton was chosen as the case study for implementing Google-OR Tools solver (Google, 2025) and testing its performance on complex VRPs with a large dataset.

The dataset for Novo Sarajevo was partially provided by the company responsible for the waste collection service. The map used as the basis for data collection is available through the publicly available online platform (Upravljanje otpadom (Waste management). (n.d.)), which is primarily intended to inform citizens about the collection schedules of their household waste containers. As shown on Figure 5, each macro-point (i.e., the location of several containers grouped together) is represented on the city map as a small red circle, covering the area in which the containers are placed. By selecting each circle on the map, the following relevant information could be retrieved:

the geographic coordinates of the container, its capacity and collection frequency and the type of the waste collected. In this study, only containers designated for mixed/household waste were considered. Depots are marked separately on the map with a small recycle sign, however the only relevant depot for this study is the one located slightly on the outskirts of the city. The container data were collected manually from the pop-up information of each map point, which included the weekly waste collection schedule, the container/bin capacity (in litres), and the coordinates. The vehicle data required to complete the instance file were taken from the documentation provided by the municipality (Municipality of Novo Sarajevo, 2016-2021). However, this file is already nine years old, so it remains unclear how many vehicles are in active use now. The reported vehicle capacities were listed in tonnes, therefore to ensure compatibility with the container data expressed in litres, a conversion from tonnes to litres was applied, using density values for mixed household waste obtained from an online conversion tool (Aqua-Calc, n.d.).

Figure 5 The City Map with Macro-Points

The possible visiting schedules were constructed as follows. The listed collection frequency for each container in the map available had to be respected. However, the exact visiting schedule is selected through the optimization in the scheduling part of the problem. When creating visiting schedules, the prerequisite was that at least one day between consecutive visits was required, except in cases when containers had to be emptied each day in a week. Thus, in a situation where the container was originally

scheduled for Tuesday, Thursday and Saturday, when applying the rule, four possible visiting combinations were generated.

The distances between containers and the depot were determined using their geographic coordinates. For each location pair the *OpenRouteService* tool (Heidelberg Institute for Geoinformation Technology [HeiGIT], n.d.) was used to calculate driving distances, providing a more realistic representation of the dataset.

I	***n***	***m***	***t***	***Q***
ns01	497	7	7	**15000**

Table 4 Real-Life Instance Characteristics

On Table 4 characteristics of the instance are presented, while on Table 5, the structure of the instance *ns01* is described. The complete instance is available in the Appendix I.

Header Row			
Type (type)	**Number of Vehicles (m)**	**Number of Containers (n)**	**Number of Days (t)**
1	7	497	7
Duration and Capacity Constraints			
Maximum Duration of Route per Day (D)		**Maximum Vehicle Capacity per day (Q)**	
0		15000	
Customer Information			

Container ID (i)	**X Coordinate (x)**	**Y Coordinate (y)**	**Service Duration (d)**	**Volume (d)**	**Frequency of Visit (f)**	**Number of Combinations**	**List of Possible Visit Combinations (in decimal)**
0	43.863417	18.346001	0	0	0		
1	43.870030	18.387655	0	1100	3	4	21, 37, 41, 42
40	43.862487	18.388091	0	1100	6	7	63, 95, 111, 119, 123, 125, 126

Table 5 Real-Life Input File Structure

7.2.1 Results

To illustrate the capability of the solver, the results of a real-life instance *ns01* are presented in this chapter.

The results presented were obtained within a computational time of exactly 12 hours for scheduling and 1710 seconds per day for routing. For the generation of the initial solution, solver used *Path Cheapest Arc* heuristic, while for the improvement *Guided Local Search* metaheuristic is applied. Consequently, presented results might not be optimal, as other provided algorithms could potentially yield even better results.

Table 7 summarizes the results on a daily level within a planning horizon of a week. It provides the number of visited containers, the number of utilized vehicles, the number of constructed routes, as well as total distance travelled across multiple days. This table demonstrates that the solver with its parameters was able to consider the visibility patterns and service frequencies and to translate them to routes which successfully balance workload across the planning horizon.

Day	***Containers Visited***	***Total Travelled Distance***	***Waste Collected***	***Vehicles Used***	***Total Trips***
Monday	294	307.01	301,900	6	22
Tuesday	285	300.42	287,700	6	20
Wednesday	306	298.93	312,520	7	21
Thursday	286	301.48	287,080	6	20
Friday	298	310.62	305,440	6	22
Saturday	280	283.88	283,060	7	19
Sunday	110	105.91	109,820	5	7

Table 6 Execution Summary for the 7 days Planning Horizon

To provide a more detailed overview, each day is further evaluated separately. For each day in the planning horizon, the set of generated routes is visualized on a map (Figures 6-12), providing a clearer interpretation of how waste collections are geographically distributed. Alongside visualization, similarly to Table 6, Tables 7-13 include route-specific information, including the vehicle ID, the number of containers visited, the volume collected, the number of routes generated and the total travelled distance.

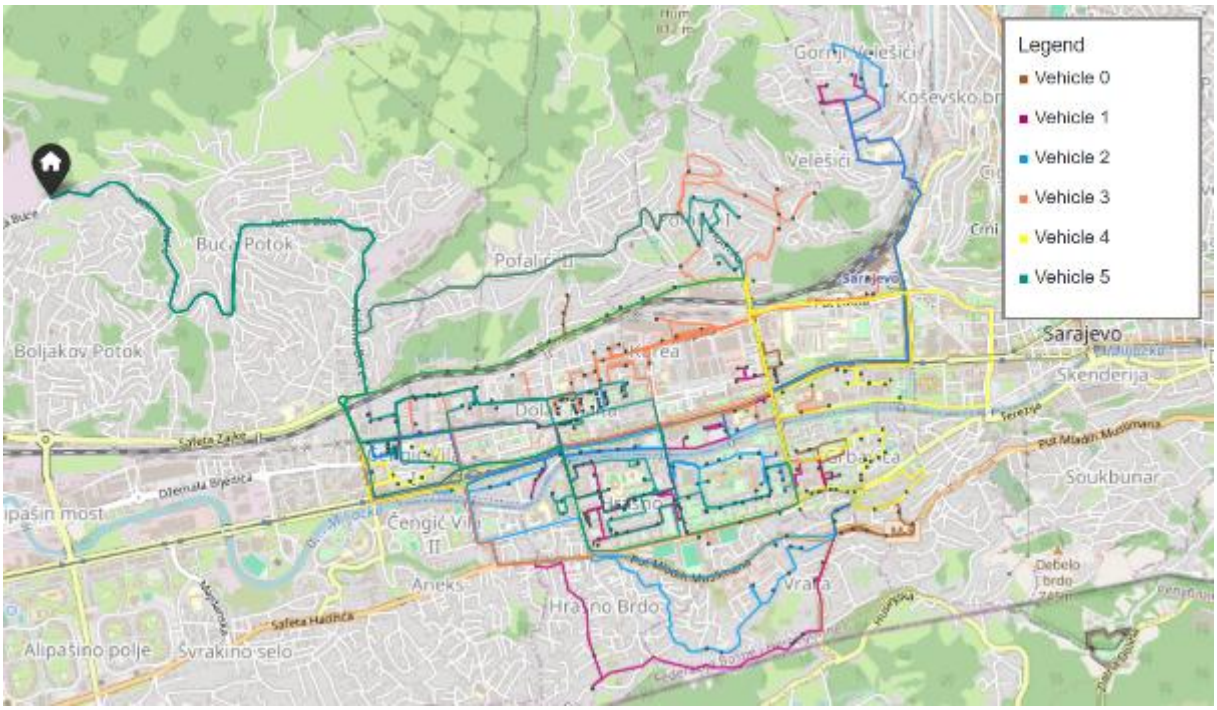

Figure 6 Route Visualization for Day 1

MONDAY

Vehicle	*Routes*	*Volume Collected*	*Capacity Utilized*	*Distance (km)*
0	3	34,580	76.80%	41.27
1	4	58,160	97.00%	56.54
2	4	58,160	97.00%	56.28
3	4	56,680	94.50%	58.24
4	3	37,500	83.30%	41.5
5	4	56,820	94.70%	53.18
6	0	0	0.00%	0

Table 7 Day 1 Routing Performance

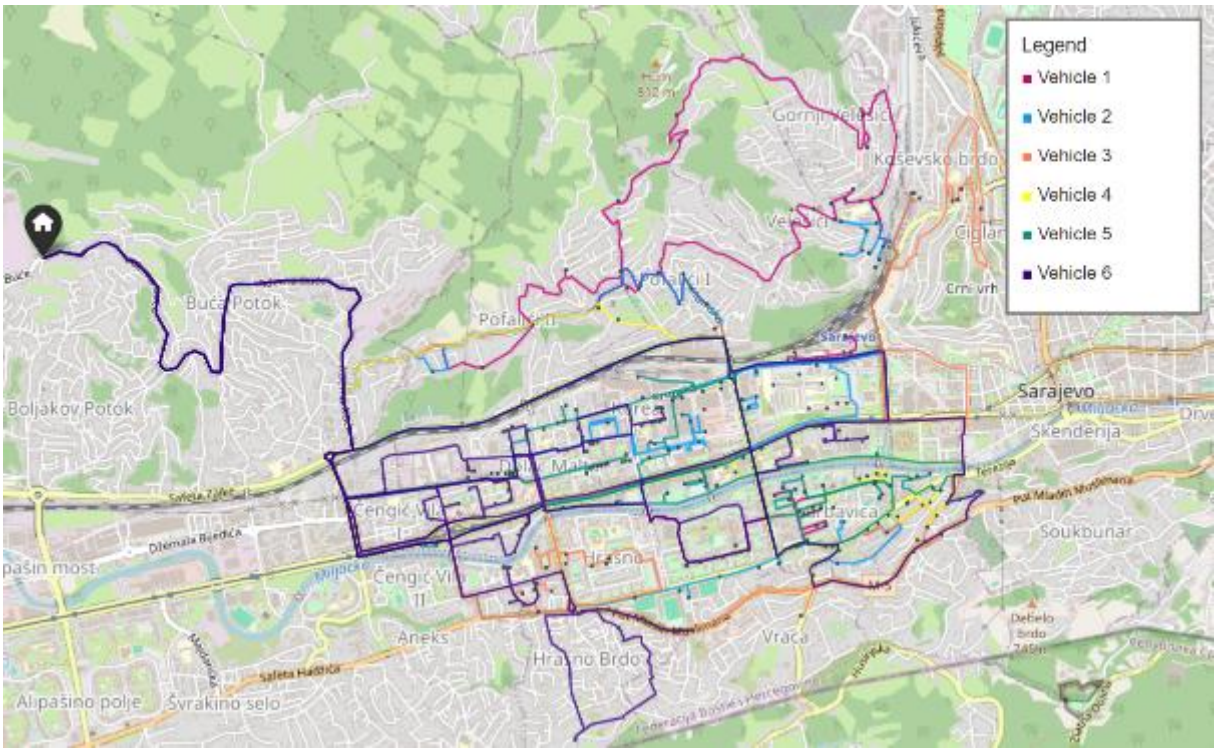

Figure 7 Route Visualization for Day 2

TUESDAY

Vehicle	Routes	Volume Collected	Capacity Utilized	Distance (km)
0	0	0	0.00%	0
1	4	57,920	96.50%	61.55
2	4	57,920	96.50%	60.88
3	2	29,560	98.50%	31.14
4	1	14,780	98.50%	16.92
5	4	58,500	97.50%	58.39
6	5	69,020	96.70%	71.54

Table 8 Day 2 Routing Performance

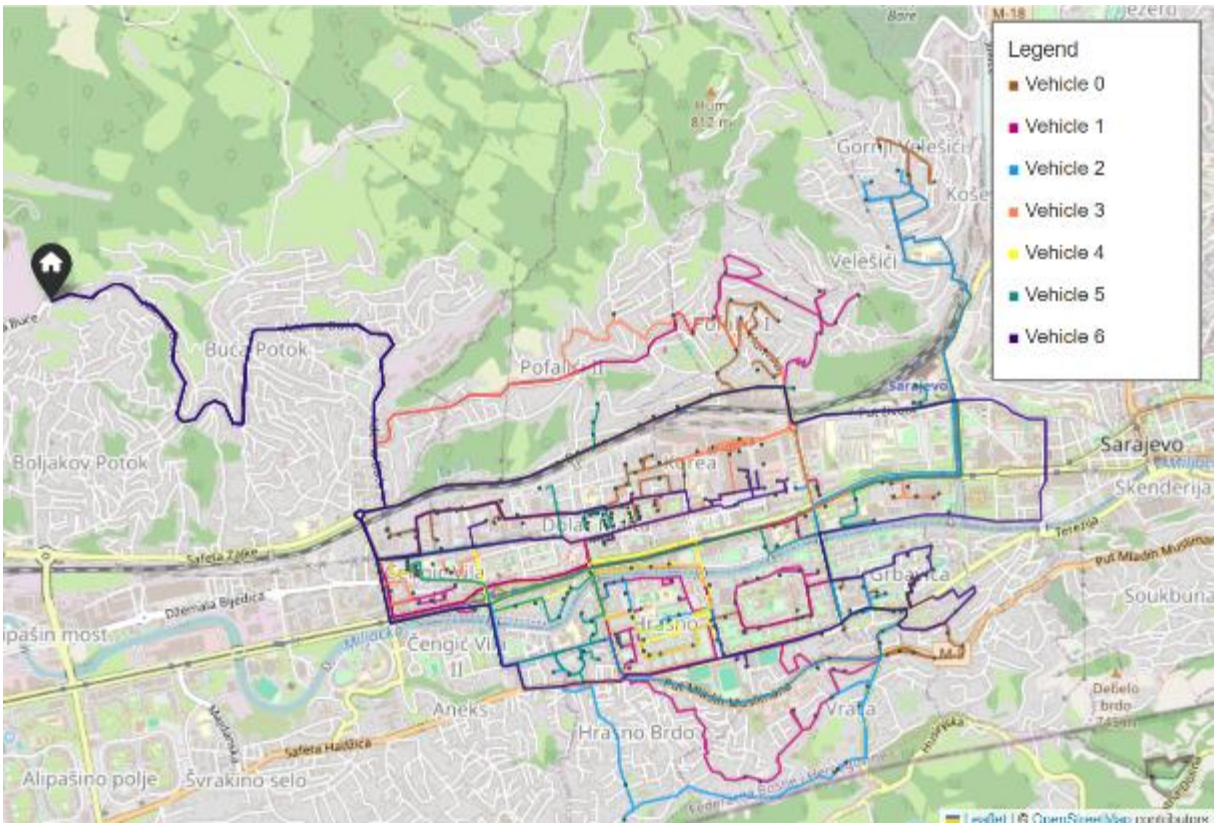

Figure 8 Route Visualization for Day 3

WEDNESDAY

Vehicle	Routes	Volume Collected	Capacity Utilized	Distance (km)
0	4	57,780	96.30%	55.17
1	5	67,200	89.60%	69.05
2	2	29,320	97.70%	31.69
3	3	42,280	93.90%	41.02
4	1	13,440	89.60%	13.48
5	3	44,100	98.00%	40.93
6	3	43,860	97.50%	47.59

Table 9 Day 3 Routing Performance

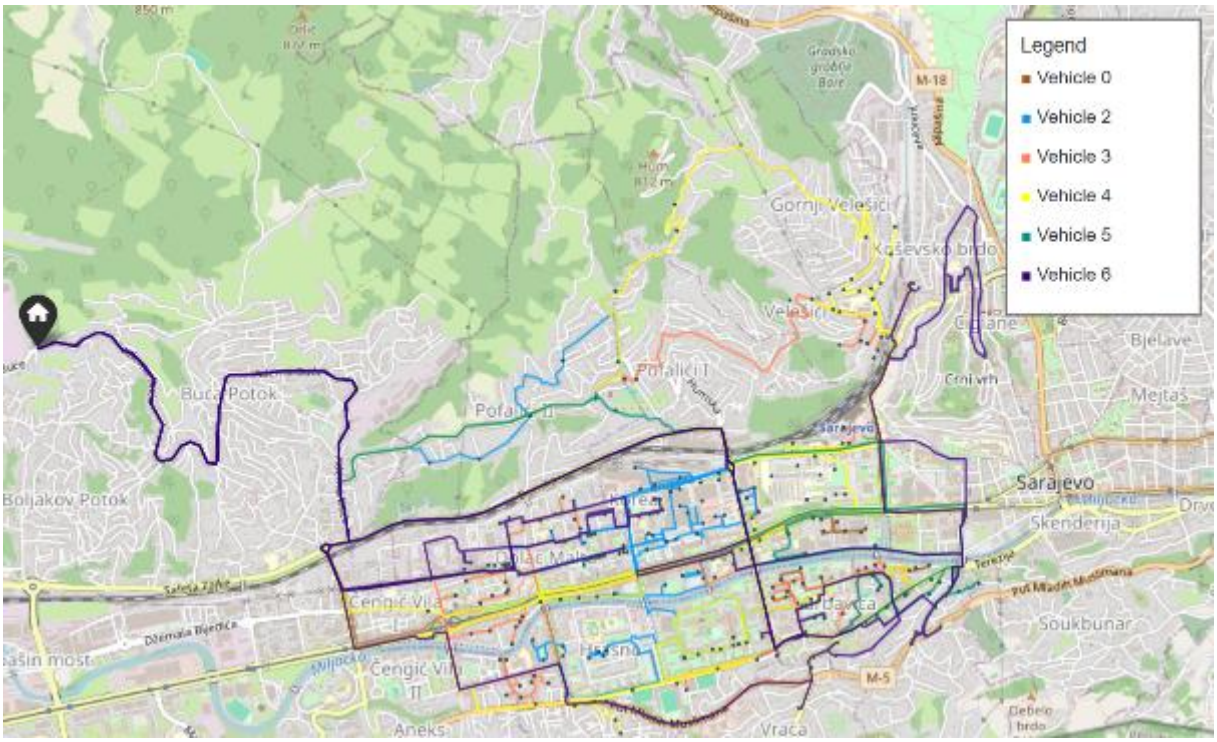

Figure 9 Route Visualization for Day 4

THURSDAY

Vehicle	Routes	Volume Collected	Capacity Utilized	Distance (km)
0	2	28,460	94.90%	31.28
1	0	0	0.00%	0
2	5	69,160	92.20%	69.4
3	4	58,500	97.50%	59.34
4	4	57,780	96.30%	66.7
5	1	14,780	98.50%	17.25
6	4	58,400	97.30%	57.51

Table 10 Day 4 Routing Performance

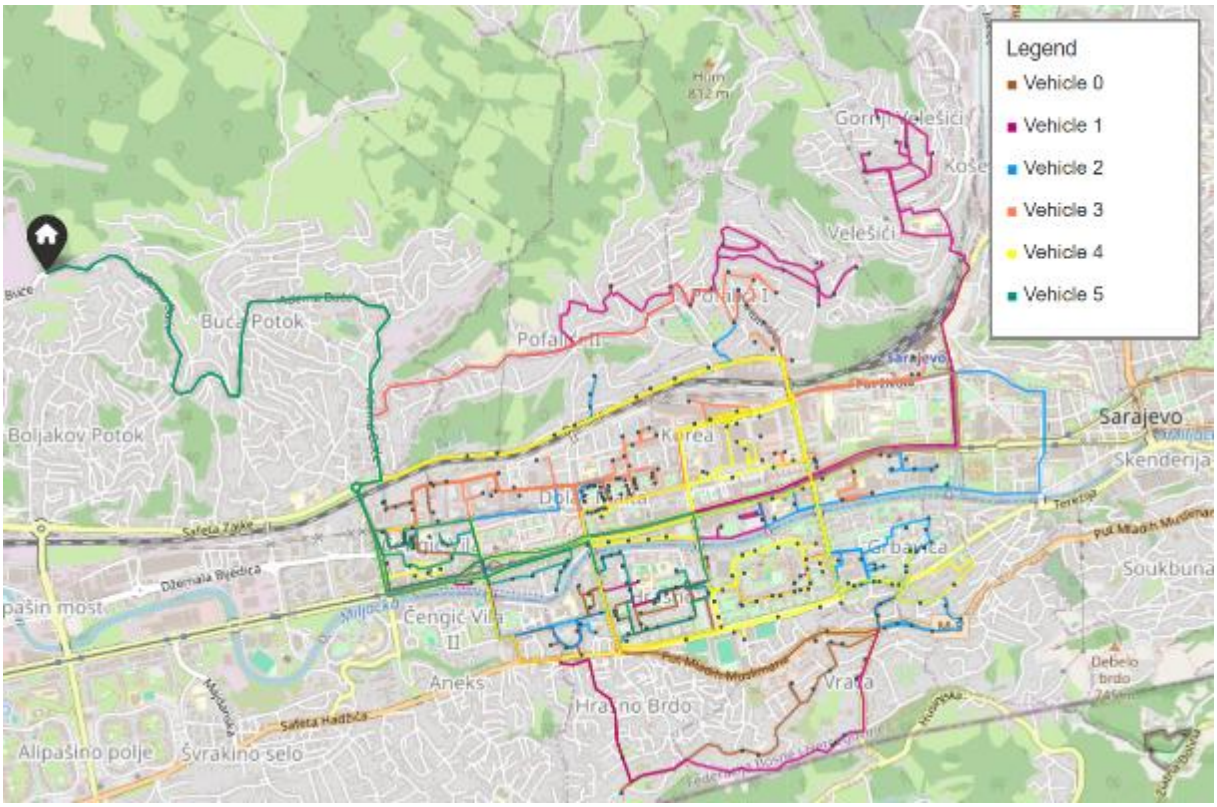

Figure 10 Route Visualization for Day 5

FRIDAY

Vehicle	Routes	Volume Collected	Capacity Utilized	Distance (km)
0	4	45,200	75.30%	51
1	3	43,140	95.90%	45.35
2	4	58,160	97.00%	61.73
3	5	71,220	95.00%	67.78
4	4	58,400	97.30%	60.04
5	2	29,320	97.70%	24.72
6	0	0	0.00%	0

Table 11 Day 5 Routing Performance

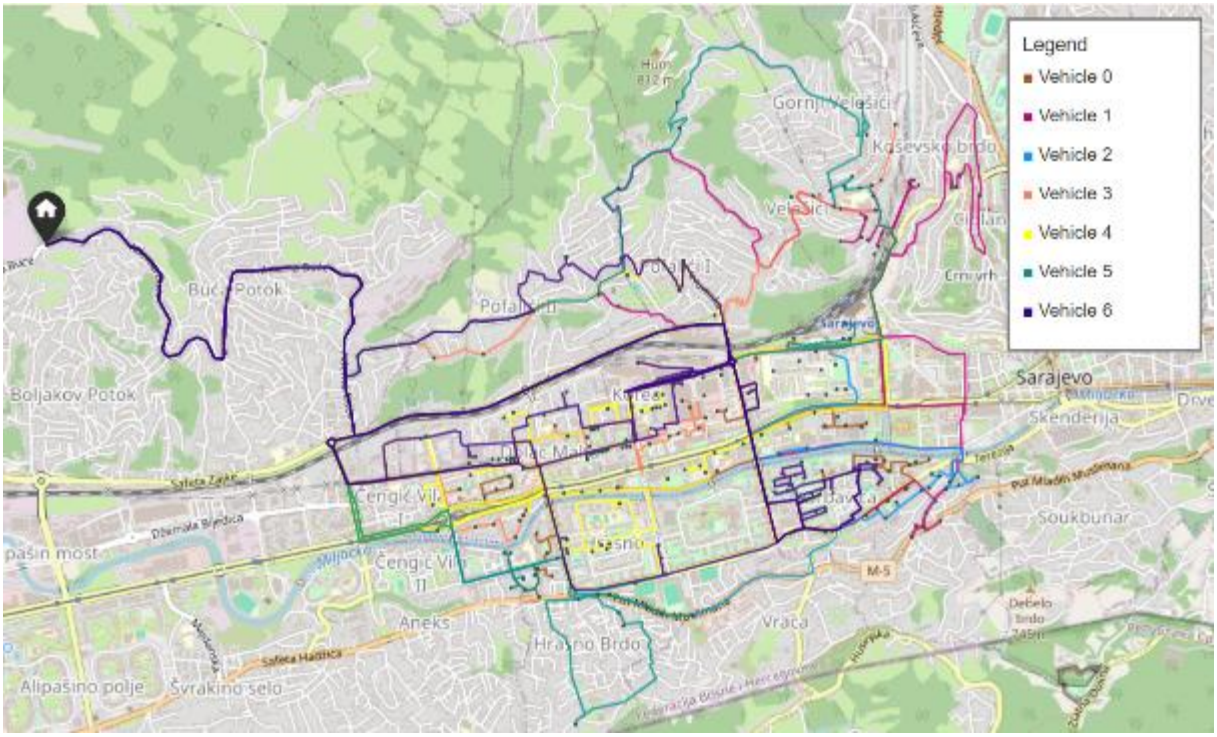

Figure 11 Route Visualization for Day 6

SATURDAY

Vehicle	Routes	Volume Collected	Capacity Utilized	Distance (km)
0	4	58,160	97.00%	64.67
1	4	53,140	88.60%	54.24
2	1	14,160	94.40%	17.9
3	1	14,780	98.50%	17.83
4	3	43,620	97.40%	41.74
5	2	27,980	93.30%	28.62
6	4	56,680	94.50%	58.88

Table 12 Day 6 Routing Performance

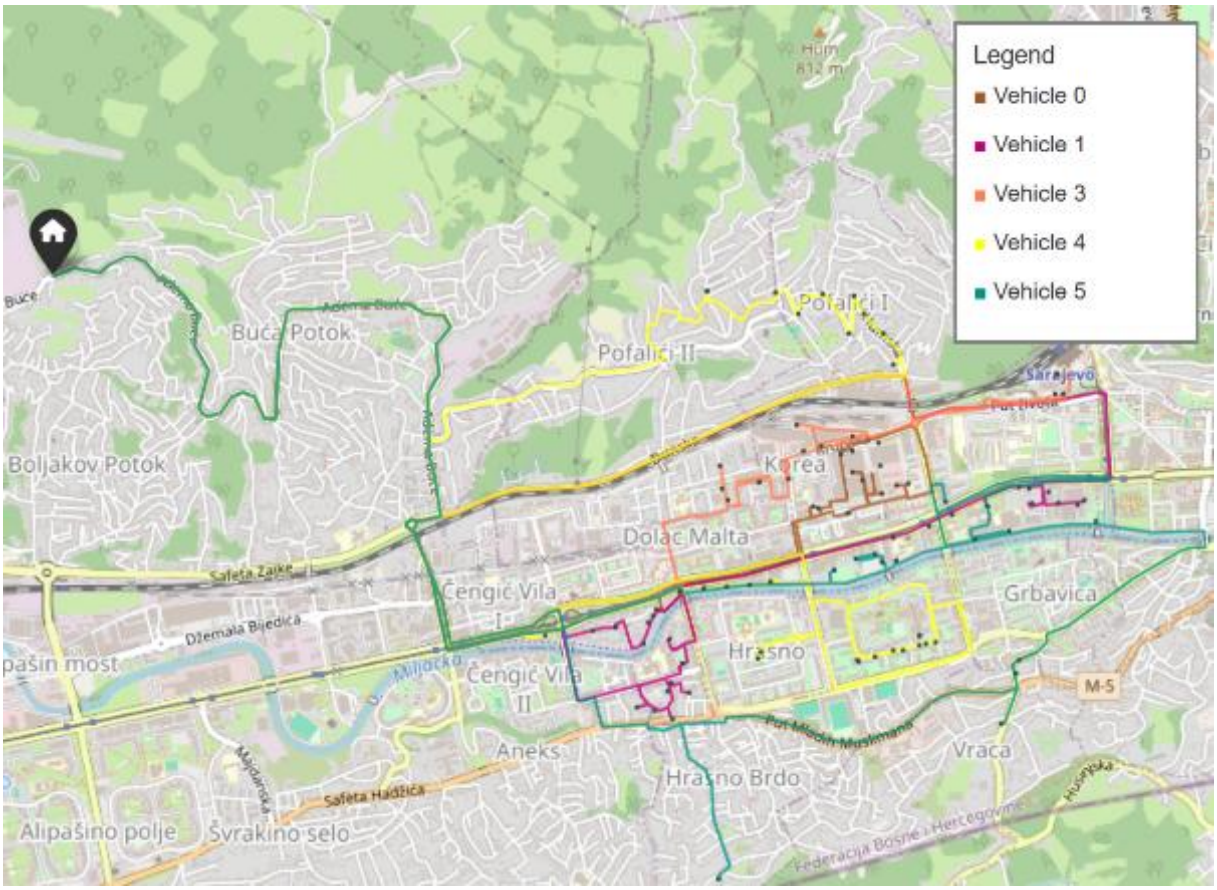

Figure 12 Route Visualization for Day 7

SUNDAY				
Vehicle	***Routes***	***Volume Collected***	***Capacity Utilized***	***Distance (km)***
0	1	14,300	95.30%	15.05
1	2	26,020	86.70%	28.16
2	0	0	0.00%	0
3	1	13,060	87.10%	15.98
4	2	27,980	93.30%	26.12
5	1	14,780	98.50%	20.6
6	0	0	0.00%	0

Table 13 Day 7 Routing Performance

Parameter Tuning Search Space Time Limit

Since the quality of the solution of the VRPs often depends on the search time limit setup, different limits were tested, ranging from 30 seconds up to 1800, increasing for 30 seconds in each subsequent case. To make the results comparable, the same

schedule was used across all cases, and the objective was to identify how the quality of the solution behaves over increasing runtimes. The total distance travelled decreases from 2,243.01 km to 1,908.25 km, demonstrating an improvement of 14.9%.

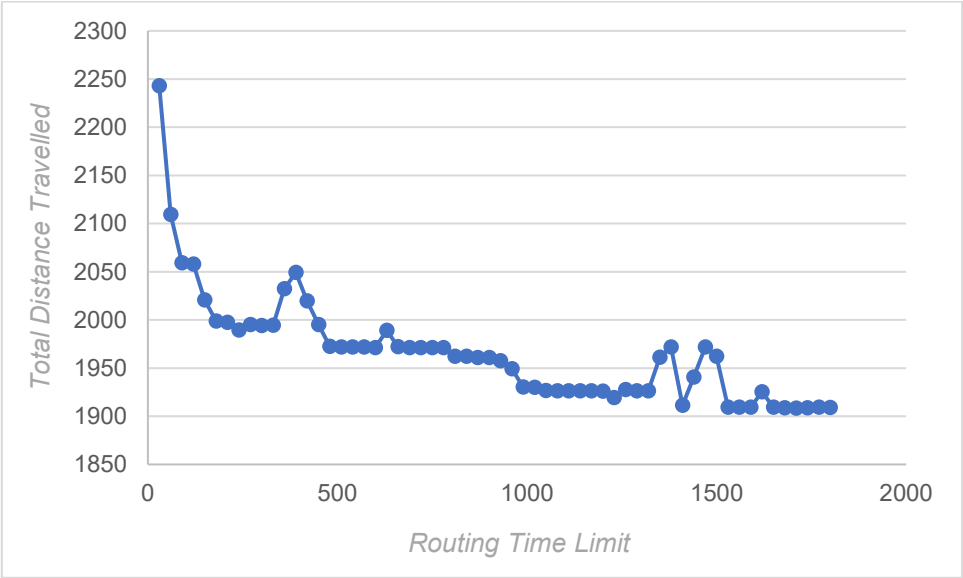

Figure 13 Time-Limit Parameter Tuning

As shown in Figure 13, four phases can be observed. Rapit improvement of the solution happens within the first 240 seconds (73.8% of the total improvement percentage), a range between 270 and 540 seconds reflect a stabilization around a local minima, a breakthrough phase between 540 and 990, where the algorithm successfully escapes local minima and reach the distance of 1,930.15 km, and the last phase of fine-tuning extending to approximately 1710 seconds, in which time-limit the best solution of 1,908.25 km is obtained. It can be observed that even the higher computational time yields slightly better results, a runtime of around 990 seconds represents a right balance between computational time and solution quality.

8. Limitations

As described in the Problem Description, due to the high complexity and lack of the data the available real case dataset is incomplete. Although in real life scenarios, complete information is rarely accessible and often changeable, the problem is significantly simplified with certain assumptions. Even though these assumptions may lead to solutions that do not fully comply with all real-world constraints, they can still serve as a starting point and a guidance for further improvements.

The following sections outline several limitations of the research, which are important to be recognized, as they might have influenced the solutions obtained.

This paper focuses on household waste collection. Since the only data provided regarding the capacity are the container sizes in litres, and vehicle capacities in tonnes, one of the limitations is that conversion from liters to kilos is done based on the calculator online provided, which might not be the accurate enough solution.

Due to the unique constraints of the waste collection problem in Sarajevo, as well as the fact that the test instances were generated directly from real-world data, no benchmark instances for the multi-trip PVRP exist in the literature. Benchmark instances are available only for the single-trip variant of the PVRP, and these were therefore used as the basis for comparative analysis.

Another limitation is that routes are not generated in the scheduling phase. The reason behind this decision is in the computational complexity, which results in unmanageably large search spaces.

The distance matrix is calculated by summing up smaller batches in which the distances were generated. Since OpenRouteService supports a maximum of 2,500 distance calculations per request, but our solution requires calculating distances for 497 locations (resulting in **247,009** distance calculations), it was not possible to compute the entire distance matrix in a single run. As a result, there is a potential limitation: the distance matrix used may not be fully optimal. For instance, in some cases, the calculated distance between two locations might be slightly longer than the actual shortest route because the distances were derived from separately processed

batches. All distance calculations are based on driving routes provided by OpenRouteService.

9. Conclusion

In urban waste collection, municipalities often face the challenge of deciding when which collection point to visit and how to design the routes efficiently, as these decisions are among critical ones, directly influencing operational costs. To address this, Google OR-Tools has been chosen for solving the Multi-Trip Periodic Vehicle Routing Problem, with the aim to minimize the total distance travelled across the defined planning horizon. Since exact approaches lack solving capabilities for large scale instances, OR-Tools was implemented, as known for its powerful capabilities when it comes to complex and large-scale optimization problems. Comparative analysis on Benchmark Instances was executed for the Single-Trip variation of the PVRP, while a real-world instance demonstrated the solver's practical relevance and its applicability in the real-world problems.

In this thesis, OR-Tools optimization capabilities proved that they provide a powerful framework for modeling and solving complex real-life optimization problems such as the multi-trip periodic capacitated vehicle routing problem. The configuration process, while requiring background knowledge related to solver parameters such as callbacks, dimensions, and penalties, is mostly straightforward compared to other optimization approaches. OR-Tools is still actively developed, and its continuous improvement indicates that it is on a promising path to become more used in both academia and industry.

The solver provides very good results on Benchmark Instances, matching for several instances the best known solutions from the literature. The obtained results on the real-world instance confirm that realistic routing and scheduling scenarios can be effectively modeled using the solver, but there remains significant potential for further extensions. Future work could incorporate heterogeneous vehicle fleets, which would more accurately reflect the actual composition of the waste management fleet, time windows, and additional parameters such as vehicle speeds, load/unload times, and container-specific handling requirements. These additions would enable the definition of more precise penalties and constraints, thereby yielding solutions that better reflect real-world operations and enhance the practical applicability of the approach.

This research was done only for one district of Sarajevo. Because of that, the results are limited to a smaller area of the city. For further research, the model could be

expanded to all seven districts, and in that way cover the whole city. This would make it possible to test how the solution works with bigger and more complex data, and also to compare different districts that have different density and infrastructure.

Another idea for further research lies in exploring the dedicated scheduling solvers already available in OR-Tools. While this thesis relied on custom Python code for scheduling, integrating OR-Tools scheduling solver capabilities could streamline the workflow, reduce complexity, and further improve solution quality in problems where routing and scheduling must be optimized simultaneously.

10. Bibliography

Aarts, E. H. L., Korst, J. H. M., & Laarhoven, van, P. J. M. (1997). *Simulated annealing. In E. H. L. Aarts, & J. K. Lenstra (Eds.), Local search in combinatorial optimization* (pp. 91-120). Wiley-Interscience.

Algethami, Haneen. (2023). *Local Search-Based Metaheuristic Methods for the Solid Waste Collection Problem*. Applied Computational Intelligence and Soft Computing. 2023. 1-11. https://doi.org/10.1155/2023/5398400

Amal, Belmabrouk & Lahmar, Arij & Chouikhi, Houssam & Bentaher, Hatem. (2023). Classification framework for vehicle routing problems. 161-167. 10.1109/SusTech57309.2023.10129561

A. Belmabrouk, A. Lahmar, H. Chouikhi and H. Bentaher, "*Classification framework for vehicle routing problems*," 2023 IEEE Conference on Technologies for Sustainability (SusTech), Portland, OR, USA, 2023, pp. 161-167, https://doi.org/10.1109/SusTech57309.2023.10129561

Barbucha, D.*: An agent-based guided local search for the capacited vehicle routing problem.* In: KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications. pp. 476–485. Springer (2011). https://doi.org/10.1007/978-3-642-22000-549

Beliën, J., De Boeck, L., & Van Ackere, J. (2012). *Municipal solid waste collection and management problems: A literature review. Transportation Science, 48*(1), 78–102.

Beltrami, E. J., & Bodin, L. D. (1974). *Networks and vehicle routing for municipal waste collection. Networks, 4*(1), 65–94.

Bernardino, R., Janela, J., Martins, C., Mourão, M.C., Pinto, L.S. and Rodrigues, F. (2025), *A Multi-Trip Vehicle Routing Problem With Release Dates and Interrelated Periods.* Networks, 85: 189-204. https://doi.org/10.1002/net.22258

Billy E. Gillett, Leland R. Miller, (1974) *A Heuristic Algorithm for the Vehicle-Dispatch Problem. Operations Research 22(2):*340-349.https://doi.org/10.1287/opre.22.2.340

Buhrkal, Katja & Larsen, Allan & Ropke, Stefan. (2012). *The Waste Collection Vehicle Routing Problem with Time Windows in a City Logistics Context. Procedia - Social and Behavioral Sciences. 39.* 241–254. Doi: https://doi.org/10.1016/j.sbspro.2012.03.105

Carvalho, J. M. Crespo. (2004) Logística. Lisboa, Edições Silabo, Lda., 3ª Edição

Christofides, N., & Eilon, S. (1969*). An Algorithm for the Vehicle-Dispatching Problem. OR, 20(3), 309–318.* https://doi.org/10.2307/3008733

Christofides, Nicos. (2022). *Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. Carnegie Mellon University. 3. 10*. https://doi.org/10.1007/s43069-021-00101-z.

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

Cordeau, J.-F., Gendreau, M. and Laporte, G. (1997*), A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks, 30:* 105-119. https://doi.org/10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G

Cuvelier, T., Didier, F., Furnon, V., Gay, S., Mohajeri, S., et al. (2023, February). *OR-Tools' vehicle routing solver: A generic constraint-programming solver with heuristic search for routing problems*. Paper presented at the 24e congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision (ROADEF), Rennes, France. https://hal.science/hal-04015496

David Pisinger, Stefan Ropke*, A general heuristic for vehicle routing problems, Computers & Operations Research,* Volume 34, Issue 8, 2007, Pages 2403-2435, ISSN 0305-0548, https://doi.org/10.1016/j.cor.2005.09.012

Dowd, Michael & Dixon, Anna & Kinsella, Benjamin. (2020*). Optimizing Waste Management Collection Routes in Urban Haiti: A Collaboration between DataKind and SOIL.* https://doi.org/10.48550/arXiv.2011.00303

Fred Glover*, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research,* Volume 13, Issue 5, 1986, Pages 533-549, ISSN 0305-0548, https://doi.org/10.1016/0305-0548(86)90048-1.

Furnon, V., & Perron, L. (2025, February 17). *OR-Tools Routing Library* (v9.12). Google. Retrieved from https://developers.google.com/optimization/routing/

G. Clarke, J. W. Wright, (1964) *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points.* Operations Research 12(4):568-581.

Garside, A., & Laili, N. (2019). *A Cluster-First Route-Second Heuristic Approach to Solve The Multi-Trip Periodic Vehicle Routing Problem.* Jurnal Teknik Industri, 20(2), 172-181. doi:https://doi.org/10.22219/JTIUMM.Vol20.No2.172-181

Gavrilas, Mihai. (2010). *Heuristic and metaheuristic optimization techniques with application to power systems.* International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering - Proceedings. 13-13.

Gendreau, M. (2002, July). *An introduction to Tabu Search* (Research paper). Centre de recherche sur les transports & Département d'informatique et de recherche opérationnelle, Université de Montréal. Retrieved from https://www.worldcolleges.info/sites/default/files/ai4.pdf

Genova, Kyle & Williamson, David. (2015). *An Experimental Evaluation of the Best-of-Many Christofides' Algorithm for the Traveling Salesman Problem. Lecture Notes in Computer Science.* https://doi.org/10.1007/978-3-662-48350-3_48.

Google OR-Tools. (2024, August 28). Routing Options. Retrieved from https://developers.google.com/optimization/routing/routing_options

Google. *cvrp_reload.py* (sample code). Available at: https://github.com/google/or-tools/blob/master/ortools/constraint_solver/samples/cvrp_reload.py (Accessed: 20th August 2025)

Han, H., & Ponce-Cueto, E. (2015). Waste collection vehicle routing problem: Literature review. *Promet – Traffic&Transportation, 27*(4), 345–358.

HEC Montreal. (n.d.). Data File Structure README.txt. Retrieved August 23, 2025, from https://chairelogistique.hec.ca/data/README.TXT

HeiGIT. (n.d.). *Openrouteservice*. In *OpenStreetMap Wiki*. Retrieved August 30, 2025, from https://wiki.openstreetmap.org/wiki/Openrouteservice https://doi.org/10.1287/opre.12.4.568

Hess, C., Dragomir, A. G., Doerner, K. F., & Vigo, D. (2024). Waste collection routing: a survey on problems and methods. *Central European Journal of Operations Research*, *32*(2), 399-434. https://doi.org/10.1007/s10100-023-00892-y

I. M. Chao, B. L. Golden, and E. A. Wasil, *An improved heuristic for the period vehicle routing problem. Net-works 26* (1995) 25–44

Kirkpatrick, Scott & Gelatt, C. & Vecchi, M.. (1983). *Optimization by Simulated Annealing.* Science (New York, N.Y.). 220. 671-80. https://doi.org/10.1126/science.220.4598.671.

KJKP "RAD" d.o.o. Sarajevo. (2022). *Plan poslovanja KJKP "RAD" d.o.o. Sarajevo za 2023. godinu*. Sarajevo.

Leila Mahdavi, Saeed Mansour, Mohsen Sheikh Sajadieh et al. *Sustainable multi-trip periodic redesign-routing model for municipal solid waste collection network: The case study of Tehran*, 18 March 2021, https://doi.org/10.21203/rs.3.rs-256948/v1

Lin, K., Musa, S. N., & Yap, H. J. (2023*). Adaptive Multi-Objective Algorithm for the Sustainable Electric Vehicle Routing Problem in Medical Waste Management. Transportation Research Record*, *2678*(7), 413-433. https://doi.org/10.1177/03611981231207096 (Original work published 2024)

Mattos Ribeiro, G. & Laporte, G. (2012). *An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem.* Computers and Operations Research, 39(3), 728–735. https://doi.org/10.1016/j.cor.2011.05.005

Messaoudi, B., Oulamara, A., & Salhi, S. (2023*). A decomposition approach for the periodic consistent vehicle routing problem with an application in the cleaning sector.* International Journal of Production Research, 61(22), 7727–7748. https://doi.org/10.1080/00207543.2022.2162145

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). *Equation of State calculations by fast computing machines*. *The Journal of Chemical Physics, 21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Mofid-Nakhaee E, Barzinpour F. *A multi-compartment capacitated arc routing problem with intermediate facilities for solid waste collection using hybrid adaptive large neighborhood search and whale algorithm. Waste Management & Research*. 2018;37(1):38-47. doi:10.1177/0734242X18801186

Molina, Jose & Eguia, Ignacio & Racero, Jesús. (2019). *Reducing pollutant emissions in a waste collection vehicle routing problem using a variable neighborhood tabu search algorithm: a case study*. TOP. 27. https://doi.org/10.1007/s11750-019-00505-5

Municipality of Novo Sarajevo. (2016-2021). *Plan upravljanja otpadom Općine Novo Sarajevo 2016-2021 [PDF file]*. Retrieved August 24, 2025, from https://novosarajevo.ba/userfiles/Plan_upravljanja_otpadom_Opcine_Novo_Sarajevo_2016_2021.pdf

Pacheco, Joaquín & Gonzalo-Orden, Hernán & Gómez, Jose. (2015). *A Tabu Search Method for a Bi-Objective Urban Waste Collection Problem. Computer-Aided Civil and Infrastructure Engineering*. 30. 36-53. https://doi.org/10.1111/mice.12031

Pamosoaji, Anugrah & Dewa, Parama & Krisnanta, J.V. (2019). *Proposed Modified Clarke-Wright Saving Algorithm for Capacitated Vehicle Routing Problem.* International Journal of Industrial Engineering and Engineering Management. 1. 9. https://doi.org/10.24002/ijieem.v1i1.2292.

P. Kilby, Patrick Prosser, and P. Shaw, “Guided local search for the vehicle routing problem with time windows,” in *Metaheuristics*, pp. 473–486, Springer, Berlin, Germany, 1999.

Rajni Aron, Ajith Abraham, *Resource scheduling methods for cloud computing environment: The role of meta-heuristics and artificial intelligence, Engineering Applications of Artificial Intelligence*, Volume 116, 2022,105345, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2022.105345

S. Eilon, C. D. T. Watson-Gandy, and N. Christofides, *Distribution Management: Mathematical Modelling and Practical Analysis*. Griffin, London (1971)

Shaw, P. (1998). *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems* (pp. 417–431). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-49481-2_30

Silva, A. S., Alves, F., Diaz de Tuesta, J. L., Rocha, A. M. A. C., Pereira, A. I., Silva, A. M. T., Leitão, P., & Gomes, H. T. (2021). *Solving a capacitated waste collection problem using an open-source tool*.

T. Vidal, T. Crainic, M. Gendreau, N. Lahrichi, W. Rei, *A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems,* Operations Research 60 (3) (2012) 611–624.

Toth, P. and Vigo, D. (2002) *The Vehicle Routing Problem*. Siam, Philadelphia. http://dx.doi.org/10.1137/1.9780898718515

Ufuk Dereci, Muhammed Erkan Karabekmez,*The applications of multiple route optimization heuristics and meta-heuristic algorithms to solid waste transportation: A case study in Turkey*, Decision Analytics Journal,Volume 4, 2022,100113, ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2022.100113

UN-Habitat. (2010). *Collection of municipal solid waste in developing countries*. https://unhabitat.org/sites/default/files/2021/02/2010_collection-msw-developing-countries_un-habitat.pdf
United Nations Statistics Division. (1997). **Glossary of environment statistics* (Statistical papers – Series F No. 67).* United Nations. Retrieved from https://unstats.un.org/unsd/publication/seriesf/seriesf_67e.pdf

Upravljanje otpadom [Waste management]. (n.d.). GIS Viewer. JP Rad d.o.o. Sarajevo. Retrieved August 24, 2025, from https://gis.rad.com.ba

Voudouris, C., & Tsang, E.P. (2010). Guided Local Search. *Handbook of Metaheuristics*.

Voudouris, C., & Tsang, E. (1998). *Guided Local Search and Its Application to the Traveling Salesman Problem* [PDF]. *European Journal of Operational Research.* Manuscript accepted for publication, March 1998. Retrieved from https://www.inf.ufpr.br/aurora/disciplinas/topicosia/papers/gls.pdf

Wang, C.J. & Tsang, Edward. (1991*). Solving constraint satisfaction problems using neural networks*. IEE Conference Publication. 295 - 299.

Wen, W., Fan, H., Yang, X., & Qi, T. (2015). Periodic vehicle routing problem and tabu search algorithm. *Proceedings of the 2015 International Conference on Mechatronics, Electronic, Industrial and Control Engineering* (pp. 511–514). Atlantis Press. https://doi.org/10.2991/meic-15.2015.118

Appendix

Appendix I

Real-life instance ns_01

1	**7**	**497**	**7**							
0	**15000**									
0	**15000**									
0	**15000**									
0	**15000**									
0	**15000**									
0	**15000**									
0	**15000**									
0	43.8634	18.346	0	0	0					
1	43.87	18.3877	0	1100	3	4	21	37	41	42
2	43.8688	18.3878	0	1100	3	4	21	37	41	42
3	43.87	18.4	0	1100	3	4	21	37	41	42
4	43.8685	18.4002	0	1100	3	4	21	37	41	42
5	43.8685	18.4002	0	1100	3	4	21	37	41	42
6	43.869	18.4021	0	1100	3	4	21	37	41	42
7	43.8677	18.4019	0	1100	3	4	21	37	41	42
8	43.8664	18.4012	0	1100	3	4	21	37	41	42
9	43.8661	18.4005	0	1100	3	4	21	37	41	42
10	43.866	18.3998	0	1100	3	4	21	37	41	42
11	43.8661	18.3988	0	1100	3	4	21	37	41	42
12	43.8665	18.3993	0	1100	3	4	21	37	41	42
13	43.8656	18.4007	0	1100	3	4	21	37	41	42
14	43.8656	18.4007	0	1100	3	4	21	37	41	42
15	43.8663	18.4031	0	1100	3	4	21	37	41	42
16	43.8649	18.4008	0	1100	3	4	21	37	41	42
17	43.8646	18.4001	0	1100	3	4	21	37	41	42
18	43.8635	18.3985	0	1100	3	4	21	37	41	42
19	43.865	18.3985	0	1100	3	4	21	37	41	42
20	43.8651	18.3981	0	1100	3	4	21	37	41	42
21	43.8658	18.3978	0	1100	3	4	21	37	41	42
22	43.8656	18.3967	0	1100	3	4	21	37	41	42
23	43.8655	18.3952	0	1100	3	4	21	37	41	42
24	43.8638	18.4013	0	1100	3	4	21	37	41	42
25	43.8635	18.4003	0	1100	3	4	21	37	41	42
26	43.8635	18.4003	0	1100	3	4	21	37	41	42
27	43.8636	18.3999	0	1100	3	4	21	37	41	42
28	43.8628	18.4007	0	1100	3	4	21	37	41	42
29	43.8632	18.401	0	1100	3	4	21	37	41	42
30	43.8635	18.3953	0	1100	3	4	21	37	41	42
31	43.8623	18.3939	0	1100	3	4	21	37	41	42
32	43.8625	18.393	0	1100	3	4	21	37	41	42

33	43.8619	18.393	0	1100	3	4	21	37	41	42		
34	43.8637	18.3929	0	1100	3	4	21	37	41	42		
35	43.8634	18.3908	0	1100	3	4	21	37	41	42		
36	43.8624	18.3904	0	1100	3	4	21	37	41	42		
37	43.8645	18.3871	0	1100	3	4	21	37	41	42		
38	43.863	18.389	0	1100	3	4	21	37	41	42		
39	43.8633	18.3875	0	1100	3	4	21	37	41	42		
40	43.8625	18.3881	0	1100	6	7	63	95	111	119	123	126
41	43.8625	18.3881	0	1100	6	7	63	95	111	119	123	126
42	43.8618	18.3886	0	1100	6	7	63	95	111	119	123	126
43	43.8611	18.3897	0	1100	6	7	63	95	111	119	123	126
44	43.8594	18.3913	0	1100	3	4	21	37	41	42		
45	43.8599	18.3902	0	1100	6	7	63	95	111	119	123	126
46	43.8602	18.3891	0	1100	3	4	21	37	41	42		
47	43.8611	18.3879	0	1100	6	7	63	95	111	119	123	126
48	43.8625	18.3865	0	1100	6	7	63	95	111	119	123	126
49	43.8601	18.3867	0	1100	3	4	21	37	41	42		
50	43.8618	18.3852	0	1100	6	7	63	95	111	119	123	126
51	43.8618	18.3844	0	1100	3	4	21	37	41	42		
52	43.8626	18.384	0	1100	6	7	63	95	111	119	123	126
53	43.866	18.3838	0	1100	3	4	21	37	41	42		
54	43.866	18.3838	0	1100	3	4	21	37	41	42		
55	43.8603	18.3838	0	1100	3	4	21	37	41	42		
56	43.8609	18.3825	0	1100	3	4	21	37	41	42		
57	43.8627	18.3804	0	1100	6	7	63	95	111	119	123	126
58	43.8627	18.3804	0	1100	6	7	63	95	111	119	123	126
59	43.8627	18.3804	0	1100	6	7	63	95	111	119	123	126
60	43.8601	18.3783	0	1100	3	4	21	37	41	42		
61	43.8574	18.3791	0	1100	3	4	21	37	41	42		
62	43.8579	18.3791	0	1100	3	4	21	37	41	42		
63	43.8587	18.3794	0	1100	3	4	21	37	41	42		
64	43.8581	18.3749	0	1100	3	4	21	37	41	42		
65	43.8579	18.3716	0	1100	3	4	21	37	41	42		
66	43.8552	18.3716	0	1100	3	4	21	37	41	42		
67	43.8553	18.3734	0	1100	3	4	21	37	41	42		
68	43.8561	18.3769	0	1100	3	4	21	37	41	42		
69	43.8566	18.3782	0	1100	3	4	21	37	41	42		
70	43.8578	18.3819	0	1100	3	4	21	37	41	42		
71	43.8581	18.3828	0	1100	3	4	21	37	41	42		
72	43.8586	18.3842	0	1100	3	4	21	37	41	42		
73	43.8591	18.3863	0	1100	3	4	21	37	41	42		
74	43.8595	18.3988	0	1100	7	1	127					
75	43.8595	18.3988	0	1100	7	1	127					
76	43.8588	18.3991	0	1100	7	1	127					
77	43.8588	18.3987	0	240	7	1	127					
78	43.8588	18.3954	0	1100	3	4	21	37	41	42		
79	43.8582	18.3968	0	1100	3	4	21	37	41	42		
80	43.8578	18.3929	0	1100	3	4	21	37	41	42		

81	43.8578	18.3958	0	240	3	4	21	37	41	42		
82	43.8573	18.394	0	1100	3	4	21	37	41	42		
83	43.8573	18.3972	0	1100	3	4	21	37	41	42		
84	43.8576	18.3982	0	1100	3	4	21	37	41	42		
85	43.8567	18.3929	0	1100	3	4	21	37	41	42		
86	43.8566	18.392	0	1100	3	4	21	37	41	42		
87	43.8562	18.3924	0	1100	3	4	21	37	41	42		
88	43.8562	18.3924	0	1100	3	4	21	37	41	42		
89	43.8556	18.3932	0	1100	3	4	21	37	41	42		
90	43.8553	18.3928	0	1100	3	4	21	37	41	42		
91	43.8551	18.3928	0	1100	3	4	21	37	41	42		
92	43.855	18.3928	0	1100	3	4	21	37	41	42		
93	43.855	18.3928	0	1100	3	4	21	37	41	42		
94	43.8547	18.3929	0	240	6	7	63	95	111	119	123	126
95	43.8561	18.3963	0	1100	3	4	21	37	41	42		
96	43.8562	18.3976	0	1100	3	4	21	37	41	42		
97	43.8564	18.3983	0	1100	3	4	21	37	41	42		
98	43.8564	18.3988	0	1100	3	4	21	37	41	42		
99	43.854	18.4008	0	1100	7	1	127					
100	43.8547	18.4002	0	1100	7	1	127					
101	43.8546	18.3998	0	1100	7	1	127					
102	43.8547	18.3988	0	1100	7	1	127					
103	43.8547	18.3988	0	1100	7	1	127					
104	43.8551	18.3983	0	1100	7	1	127					
105	43.8547	18.3974	0	1100	7	1	127					
106	43.8552	18.3973	0	1100	6	7	63	95	111	119	123	126
107	43.8552	18.3968	0	1100	7	1	127					
108	43.8536	18.3962	0	1100	7	1	127					
109	43.8545	18.3951	0	1100	7	1	127					
110	43.8542	18.3933	0	1100	7	1	127					
111	43.8533	18.3944	0	240	7	1	127					
112	43.8538	18.3921	0	1100	7	1	127					
113	43.8531	18.3916	0	1100	7	1	127					
114	43.8533	18.3902	0	1100	7	1	127					
115	43.8525	18.3903	0	1100	7	1	127					
116	43.8524	18.3896	0	1100	7	1	127					
117	43.8527	18.3892	0	1100	7	1	127					
118	43.8525	18.3882	0	1100	7	1	127					
119	43.8521	18.3871	0	1100	7	1	127					
120	43.8517	18.3838	0	1100	7	1	127					
121	43.8516	18.3826	0	1100	7	1	127					
122	43.8515	18.3815	0	1100	7	1	127					
123	43.8514	18.3802	0	1100	7	1	127					
124	43.8513	18.3789	0	1100	7	1	127					
125	43.8506	18.3778	0	1100	7	1	127					
126	43.8504	18.3775	0	1100	7	1	127					
127	43.8501	18.3773	0	1100	7	1	127					
128	43.8493	18.3771	0	1100	7	1	127					

129	43.8499	18.3756	0	1100	7	1	127					
130	43.8498	18.3744	0	1100	7	1	127					
131	43.8496	18.3719	0	240	7	1	127					
132	43.855	18.3915	0	1100	3	4	21	37	41	42		
133	43.8554	18.3909	0	1100	6	7	63	95	111	119	123	126
134	43.8571	18.3906	0	1100	3	4	21	37	41	42		
135	43.8573	18.3906	0	1100	3	4	21	37	41	42		
136	43.8553	18.3903	0	1100	6	7	63	95	111	119	123	126
137	43.8547	18.3903	0	1100	3	4	21	37	41	42		
138	43.857	18.3894	0	1100	6	7	63	95	111	119	123	126
139	43.8561	18.3895	0	1100	6	7	63	95	111	119	123	126
140	43.855	18.3895	0	1100	6	7	63	95	111	119	123	126
141	43.8545	18.3891	0	1100	3	4	21	37	41	42		
142	43.8555	18.3885	0	1100	6	7	63	95	111	119	123	126
143	43.8566	18.388	0	1100	6	7	63	95	111	119	123	126
144	43.8558	18.388	0	1100	6	7	63	95	111	119	123	126
145	43.8565	18.3874	0	1100	6	7	63	95	111	119	123	126
146	43.8543	18.3878	0	1100	6	7	63	95	111	119	123	126
147	43.8541	18.3872	0	240	3	4	21	37	41	42		
148	43.8556	18.3869	0	1100	3	4	21	37	41	42		
149	43.8561	18.3868	0	1100	3	4	21	37	41	42		
150	43.8561	18.3863	0	1100	3	4	21	37	41	42		
151	43.8555	18.3863	0	1100	3	4	21	37	41	42		
152	43.8555	18.3866	0	1100	3	4	21	37	41	42		
153	43.8555	18.3861	0	1100	3	4	21	37	41	42		
154	43.8538	18.3859	0	240	3	4	21	37	41	42		
155	43.8544	18.3862	0	1100	6	7	63	95	111	119	123	126
156	43.8545	18.386	0	1100	6	7	63	95	111	119	123	126
157	43.8553	18.3854	0	1100	3	4	21	37	41	42		
158	43.8556	18.3847	0	240	6	7	63	95	111	119	123	126
159	43.8572	18.388	0	1100	6	7	63	95	111	119	123	126
160	43.8568	18.3864	0	1100	6	7	63	95	111	119	123	126
161	43.8576	18.3851	0	1100	6	7	63	95	111	119	123	126
162	43.8566	18.3854	0	1100	6	7	63	95	111	119	123	126
163	43.8561	18.3831	0	1100	3	4	21	37	41	42		
164	43.8564	18.3827	0	240	3	4	21	37	41	42		
165	43.8563	18.382	0	1100	3	4	21	37	41	42		
166	43.856	18.3811	0	240	6	7	63	95	111	119	123	126
167	43.8562	18.3804	0	1100	3	4	21	37	41	42		
168	43.8555	18.3803	0	1100	3	4	21	37	41	42		
169	43.8555	18.3798	0	1100	3	4	21	37	41	42		
170	43.8555	18.3831	0	1100	6	7	63	95	111	119	123	126
171	43.8557	18.3816	0	1100	3	4	21	37	41	42		
172	43.8555	18.3831	0	1100	6	7	63	95	111	119	123	126
173	43.8548	18.3815	0	1100	6	7	63	95	111	119	123	126
174	43.8552	18.3811	0	240	6	7	63	95	111	119	123	126
175	43.855	18.3805	0	1100	3	4	21	37	41	42		
176	43.8546	18.3796	0	1100	3	4	21	37	41	42		

177	43.8552	18.379	0	240	3	4	21	37	41	42
178	43.8551	18.3779	0	1100	3	4	21	37	41	42
179	43.8539	18.3844	0	1100	3	4	21	37	41	42
180	43.8546	18.3832	0	1100	3	4	21	37	41	42
181	43.854	18.383	0	1100	3	4	21	37	41	42
182	43.8543	18.3829	0	1100	3	4	21	37	41	42
183	43.8543	18.3829	0	1100	3	4	21	37	41	42
184	43.8545	18.3828	0	1100	3	4	21	37	41	42
185	43.8545	18.3824	0	1100	3	4	21	37	41	42
186	43.8545	18.3822	0	1100	3	4	21	37	41	42
187	43.8544	18.382	0	1100	3	4	21	37	41	42
188	43.8542	18.3817	0	1100	3	4	21	37	41	42
189	43.8541	18.3818	0	1100	3	4	21	37	41	42
190	43.8541	18.3818	0	1100	3	4	21	37	41	42
191	43.8539	18.3818	0	1100	3	4	21	37	41	42
192	43.8538	18.3819	0	1100	3	4	21	37	41	42
193	43.8536	18.382	0	1100	3	4	21	37	41	42
194	43.8536	18.3815	0	1100	3	4	21	37	41	42
195	43.8537	18.3814	0	1100	3	4	21	37	41	42
196	43.8539	18.3814	0	1100	3	4	21	37	41	42
197	43.8542	18.3813	0	1100	3	4	21	37	41	42
198	43.8542	18.3809	0	1100	3	4	21	37	41	42
199	43.8542	18.3807	0	1100	3	4	21	37	41	42
200	43.8541	18.3805	0	1100	3	4	21	37	41	42
201	43.8539	18.3802	0	1100	3	4	21	37	41	42
202	43.8537	18.3803	0	1100	3	4	21	37	41	42
203	43.8534	18.3804	0	1100	3	4	21	37	41	42
204	43.8536	18.381	0	1100	3	4	21	37	41	42
205	43.8533	18.3799	0	1100	3	4	21	37	41	42
206	43.8534	18.3799	0	1100	3	4	21	37	41	42
207	43.8536	18.3798	0	1100	3	4	21	37	41	42
208	43.8539	18.3797	0	1100	3	4	21	37	41	42
209	43.8539	18.3794	0	1100	3	4	21	37	41	42
210	43.8539	18.3791	0	1100	3	4	21	37	41	42
211	43.8538	18.3789	0	1100	3	4	21	37	41	42
212	43.8536	18.3786	0	1100	3	4	21	37	41	42
213	43.8535	18.3787	0	1100	3	4	21	37	41	42
214	43.8532	18.3788	0	1100	3	4	21	37	41	42
215	43.8537	18.3846	0	1100	3	4	21	37	41	42
216	43.8536	18.3843	0	1100	3	4	21	37	41	42
217	43.8536	18.3841	0	1100	3	4	21	37	41	42
218	43.8535	18.3832	0	1100	3	4	21	37	41	42
219	43.8535	18.3829	0	1100	3	4	21	37	41	42
220	43.8534	18.3827	0	1100	3	4	21	37	41	42
221	43.8534	18.3825	0	1100	3	4	21	37	41	42
222	43.8533	18.3822	0	1100	3	4	21	37	41	42
223	43.8533	18.3819	0	1100	3	4	21	37	41	42
224	43.853	18.3817	0	1100	3	4	21	37	41	42

225	43.8525	18.3798	0	240	3	4	21	37	41	42
226	43.8529	18.3801	0	1100	3	4	21	37	41	42
227	43.8529	18.3799	0	1100	3	4	21	37	41	42
228	43.8528	18.3796	0	1100	3	4	21	37	41	42
229	43.8528	18.3794	0	1100	3	4	21	37	41	42
230	43.8528	18.3792	0	1100	3	4	21	37	41	42
231	43.8527	18.379	0	1100	3	4	21	37	41	42
232	43.8553	18.3769	0	1100	3	4	21	37	41	42
233	43.855	18.3763	0	1100	3	4	21	37	41	42
234	43.855	18.3758	0	240	3	4	21	37	41	42
235	43.8538	18.3745	0	1100	3	4	21	37	41	42
236	43.8545	18.3735	0	1100	3	4	21	37	41	42
237	43.8531	18.3767	0	240	3	4	21	37	41	42
238	43.8531	18.3769	0	240	3	4	21	37	41	42
239	43.853	18.3767	0	1100	3	4	21	37	41	42
240	43.8531	18.3764	0	1100	3	4	21	37	41	42
241	43.8531	18.3759	0	1100	3	4	21	37	41	42
242	43.853	18.3755	0	1100	3	4	21	37	41	42
243	43.8535	18.3748	0	1100	3	4	21	37	41	42
244	43.8532	18.3744	0	1100	3	4	21	37	41	42
245	43.8528	18.374	0	1100	3	4	21	37	41	42
246	43.8536	18.3733	0	1100	3	4	21	37	41	42
247	43.8535	18.3731	0	1100	3	4	21	37	41	42
248	43.8536	18.3725	0	240	3	4	21	37	41	42
249	43.8526	18.3776	0	1100	3	4	21	37	41	42
250	43.8522	18.3766	0	1100	3	4	21	37	41	42
251	43.8518	18.3768	0	1100	3	4	21	37	41	42
252	43.8517	18.3767	0	1100	3	4	21	37	41	42
253	43.8516	18.3757	0	1100	3	4	21	37	41	42
254	43.8513	18.375	0	1100	3	4	21	37	41	42
255	43.8523	18.374	0	240	3	4	21	37	41	42
256	43.8516	18.3733	0	1100	3	4	21	37	41	42
257	43.8516	18.3733	0	240	3	4	21	37	41	42
258	43.8516	18.3733	0	1100	3	4	21	37	41	42
259	43.8542	18.3711	0	1100	3	4	21	37	41	42
260	43.854	18.3707	0	1100	3	4	21	37	41	42
261	43.8541	18.3703	0	1100	3	4	21	37	41	42
262	43.8539	18.3696	0	1100	3	4	21	37	41	42
263	43.854	18.3694	0	1100	3	4	21	37	41	42
264	43.8527	18.3696	0	1100	3	4	21	37	41	42
265	43.8523	18.3698	0	1100	3	4	21	37	41	42
266	43.8523	18.3717	0	1100	3	4	21	37	41	42
267	43.8513	18.3705	0	240	3	4	21	37	41	42
268	43.8514	18.3701	0	1100	3	4	21	37	41	42
269	43.8517	18.3693	0	1100	3	4	21	37	41	42
270	43.8532	18.3682	0	240	3	4	21	37	41	42
271	43.8529	18.3683	0	1100	3	4	21	37	41	42
272	43.853	18.3672	0	1100	3	4	21	37	41	42

273	43.8532	18.3665	0	1100	3	4	21	37	41	42		
274	43.8531	18.3665	0	1100	3	4	21	37	41	42		
275	43.8528	18.3666	0	1100	3	4	21	37	41	42		
276	43.852	18.3682	0	240	3	4	21	37	41	42		
277	43.8518	18.3665	0	240	3	4	21	37	41	42		
278	43.8518	18.3677	0	1100	3	4	21	37	41	42		
279	43.8516	18.3678	0	1100	3	4	21	37	41	42		
280	43.8515	18.3678	0	1100	3	4	21	37	41	42		
281	43.8515	18.3683	0	1100	3	4	21	37	41	42		
282	43.8514	18.3683	0	1100	3	4	21	37	41	42		
283	43.8512	18.3684	0	1100	3	4	21	37	41	42		
284	43.8508	18.3716	0	240	3	4	21	37	41	42		
285	43.8507	18.3707	0	1100	3	4	21	37	41	42		
286	43.8502	18.37	0	1100	3	4	21	37	41	42		
287	43.8501	18.3692	0	1100	3	4	21	37	41	42		
288	43.85	18.3685	0	1100	3	4	21	37	41	42		
289	43.8504	18.3681	0	240	3	4	21	37	41	42		
290	43.8505	18.3667	0	1100	3	4	21	37	41	42		
291	43.8486	18.3763	0	1100	3	4	21	37	41	42		
292	43.8469	18.3767	0	240	6	7	63	95	111	119	123	126
293	43.8486	18.3767	0	1100	3	4	21	37	41	42		
294	43.8495	18.3784	0	1100	7	1	127					
295	43.8491	18.379	0	1100	3	4	21	37	41	42		
296	43.8486	18.3791	0	1100	7	1	127					
297	43.8484	18.3798	0	1100	6	7	63	95	111	119	123	126
298	43.8478	18.3785	0	1100	6	7	63	95	111	119	123	126
299	43.8475	18.3794	0	1100	6	7	63	95	111	119	123	126
300	43.8475	18.3783	0	1100	6	7	63	95	111	119	123	126
301	43.8478	18.3785	0	1100	6	7	63	95	111	119	123	126
302	43.8474	18.3776	0	1100	3	4	21	37	41	42		
303	43.8468	18.3776	0	240	6	7	63	95	111	119	123	126
304	43.8465	18.3785	0	240	6	7	63	95	111	119	123	126
305	43.8508	18.3799	0	240	3	4	21	37	41	42		
306	43.8509	18.3804	0	1100	3	4	21	37	41	42		
307	43.851	18.3809	0	1100	3	4	21	37	41	42		
308	43.8504	18.3811	0	1100	3	4	21	37	41	42		
309	43.8504	18.3811	0	1100	3	4	21	37	41	42		
310	43.851	18.3839	0	1100	3	4	21	37	41	42		
311	43.8502	18.3843	0	1100	3	4	21	37	41	42		
312	43.8497	18.382	0	1100	3	4	21	37	41	42		
313	43.8496	18.3813	0	1100	3	4	21	37	41	42		
314	43.8492	18.3804	0	1100	3	4	21	37	41	42		
315	43.8493	18.3815	0	1100	3	4	21	37	41	42		
316	43.8488	18.3804	0	1100	3	4	21	37	41	42		
317	43.8484	18.3806	0	1100	3	4	21	37	41	42		
318	43.848	18.3808	0	1100	3	4	21	37	41	42		
319	43.8486	18.3805	0	1100	3	4	21	37	41	42		
320	43.8482	18.3807	0	1100	3	4	21	37	41	42		

321	43.8474	18.3814	0	1100	3	4	21	37	41	42		
322	43.8471	18.3813	0	1100	3	4	21	37	41	42		
323	43.849	18.3827	0	1100	3	4	21	37	41	42		
324	43.849	18.3832	0	1100	6	7	63	95	111	119	123	126
325	43.8492	18.3837	0	1100	3	4	21	37	41	42		
326	43.8494	18.3844	0	1100	3	4	21	37	41	42		
327	43.8486	18.3821	0	1100	3	4	21	37	41	42		
328	43.8487	18.3831	0	1100	6	7	63	95	111	119	123	126
329	43.8483	18.3822	0	1100	3	4	21	37	41	42		
330	43.8478	18.3822	0	1100	3	4	21	37	41	42		
331	43.8476	18.3825	0	1100	3	4	21	37	41	42		
332	43.8477	18.3832	0	1100	3	4	21	37	41	42		
333	43.8481	18.3852	0	1100	3	4	21	37	41	42		
334	43.8492	18.3863	0	1100	3	4	21	37	41	42		
335	43.849	18.3864	0	240	3	4	21	37	41	42		
336	43.8483	18.3867	0	1100	3	4	21	37	41	42		
337	43.8508	18.3863	0	1100	3	4	21	37	41	42		
338	43.8499	18.3878	0	1100	3	4	21	37	41	42		
339	43.8494	18.3872	0	1100	3	4	21	37	41	42		
340	43.8488	18.3866	0	1100	3	4	21	37	41	42		
341	43.8482	18.3869	0	1100	3	4	21	37	41	42		
342	43.8477	18.3871	0	1100	3	4	21	37	41	42		
343	43.8509	18.3883	0	1100	3	4	21	37	41	42		
344	43.851	18.3892	0	1100	3	4	21	37	41	42		
345	43.8512	18.3905	0	1100	3	4	21	37	41	42		
346	43.8501	18.3884	0	1100	3	4	21	37	41	42		
347	43.8487	18.3883	0	240	6	7	63	95	111	119	123	126
348	43.8487	18.3884	0	1100	6	7	63	95	111	119	123	126
349	43.8489	18.3892	0	1100	6	7	63	95	111	119	123	126
350	43.8491	18.3901	0	1100	6	7	63	95	111	119	123	126
351	43.8491	18.3906	0	1100	6	7	63	95	111	119	123	126
352	43.8492	18.3916	0	1100	6	7	63	95	111	119	123	126
353	43.8493	18.3919	0	1100	6	7	63	95	111	119	123	126
354	43.8494	18.3924	0	1100	6	7	63	95	111	119	123	126
355	43.8496	18.3915	0	1100	6	7	63	95	111	119	123	126
356	43.8496	18.3915	0	1100	6	7	63	95	111	119	123	126
357	43.85	18.3919	0	1100	3	4	21	37	41	42		
358	43.8514	18.3917	0	240	3	4	21	37	41	42		
359	43.8514	18.3928	0	1100	3	4	21	37	41	42		
360	43.8507	18.3923	0	1100	3	4	21	37	41	42		
361	43.8505	18.3931	0	1100	3	4	21	37	41	42		
362	43.8502	18.3923	0	1100	3	4	21	37	41	42		
363	43.8501	18.3933	0	1100	3	4	21	37	41	42		
364	43.8497	18.3925	0	1100	3	4	21	37	41	42		
365	43.8496	18.3934	0	1100	3	4	21	37	41	42		
366	43.8523	18.3946	0	1100	3	4	21	37	41	42		
367	43.852	18.3941	0	240	3	4	21	37	41	42		
368	43.8515	18.3941	0	240	3	4	21	37	41	42		

369	43.8518	18.3951	0	1100	3	4	21	37	41	42
370	43.8516	18.395	0	1100	3	4	21	37	41	42
371	43.8506	18.3942	0	240	3	4	21	37	41	42
372	43.8507	18.3956	0	1100	3	4	21	37	41	42
373	43.8496	18.3948	0	1100	3	4	21	37	41	42
374	43.8529	18.3959	0	240	3	4	21	37	41	42
375	43.8523	18.3955	0	1100	3	4	21	37	41	42
376	43.8522	18.3961	0	1100	3	4	21	37	41	42
377	43.8525	18.3973	0	1100	3	4	21	37	41	42
378	43.8528	18.3997	0	1100	3	4	21	37	41	42
379	43.8529	18.4002	0	1100	3	4	21	37	41	42
380	43.853	18.4008	0	1100	3	4	21	37	41	42
381	43.853	18.4014	0	1100	3	4	21	37	41	42
382	43.8528	18.4031	0	1100	3	4	21	37	41	42
383	43.8529	18.4035	0	1100	3	4	21	37	41	42
384	43.853	18.4043	0	1100	3	4	21	37	41	42
385	43.8525	18.4017	0	1100	3	4	21	37	41	42
386	43.8527	18.4007	0	1100	3	4	21	37	41	42
387	43.8526	18.4006	0	1100	3	4	21	37	41	42
388	43.8521	18.4012	0	1100	3	4	21	37	41	42
389	43.8519	18.401	0	1100	3	4	21	37	41	42
390	43.8525	18.3997	0	1100	3	4	21	37	41	42
391	43.852	18.3995	0	1100	3	4	21	37	41	42
392	43.8514	18.3998	0	240	3	4	21	37	41	42
393	43.8512	18.3994	0	1100	3	4	21	37	41	42
394	43.851	18.3989	0	1100	3	4	21	37	41	42
395	43.8508	18.3987	0	1100	3	4	21	37	41	42
396	43.8508	18.3987	0	1100	3	4	21	37	41	42
397	43.8504	18.3983	0	1100	3	4	21	37	41	42
398	43.8521	18.3983	0	1100	3	4	21	37	41	42
399	43.8514	18.398	0	240	3	4	21	37	41	42
400	43.8511	18.3986	0	1100	3	4	21	37	41	42
401	43.8511	18.3986	0	1100	3	4	21	37	41	42
402	43.8514	18.3971	0	1100	3	4	21	37	41	42
403	43.8514	18.3966	0	1100	3	4	21	37	41	42
404	43.8513	18.3967	0	1100	3	4	21	37	41	42
405	43.8512	18.3963	0	1100	3	4	21	37	41	42
406	43.8512	18.3957	0	1100	3	4	21	37	41	42
407	43.8511	18.396	0	240	3	4	21	37	41	42
408	43.8507	18.3975	0	1100	3	4	21	37	41	42
409	43.8505	18.3975	0	1100	3	4	21	37	41	42
410	43.8506	18.3969	0	1100	3	4	21	37	41	42
411	43.8504	18.3969	0	1100	3	4	21	37	41	42
412	43.8505	18.3964	0	1100	3	4	21	37	41	42
413	43.8503	18.3964	0	240	3	4	21	37	41	42
414	43.8501	18.3983	0	1100	3	4	21	37	41	42
415	43.8499	18.3962	0	1100	3	4	21	37	41	42
416	43.8466	18.3832	0	1100	3	4	21	37	41	42

417	43.8472	18.385	0	240	6	7	63	95	111	119	123	126
418	43.8474	18.3878	0	1100	6	7	63	95	111	119	123	126
419	43.8472	18.3885	0	1100	6	7	63	95	111	119	123	126
420	43.8482	18.3905	0	1100	6	7	63	95	111	119	123	126
421	43.8486	18.3931	0	1100	6	7	63	95	111	119	123	126
422	43.8476	18.3935	0	1100	3	4	21	37	41	42		
423	43.8487	18.3944	0	240	6	7	63	95	111	119	123	126
424	43.8495	18.3951	0	1100	6	7	63	95	111	119	123	126
425	43.8496	18.3956	0	1100	6	7	63	95	111	119	123	126
426	43.8497	18.3963	0	1100	6	7	63	95	111	119	123	126
427	43.8497	18.3967	0	1100	6	7	63	95	111	119	123	126
428	43.8496	18.3974	0	1100	6	7	63	95	111	119	123	126
429	43.8496	18.3979	0	1100	6	7	63	95	111	119	123	126
430	43.8487	18.3968	0	1100	6	7	63	95	111	119	123	126
431	43.8482	18.3966	0	1100	6	7	63	95	111	119	123	126
432	43.8488	18.3982	0	1100	3	4	21	37	41	42		
433	43.8487	18.3992	0	1100	3	4	21	37	41	42		
434	43.8486	18.4007	0	1100	3	4	21	37	41	42		
435	43.8494	18.3983	0	1100	3	4	21	37	41	42		
436	43.8504	18.4002	0	240	3	4	21	37	41	42		
437	43.8507	18.4009	0	1100	3	4	21	37	41	42		
43.8	43.8511	18.4016	0	1100	3	4	21	37	41	42		
439	43.8502	18.4015	0	240	3	4	21	37	41	42		
440	43.8496	18.4013	0	1100	3	4	21	37	41	42		
441	43.8481	18.4017	0	1100	3	4	21	37	41	42		
442	43.851	18.4022	0	1100	3	4	21	37	41	42		
443	43.8499	18.4035	0	1100	3	4	21	37	41	42		
444	43.8498	18.404	0	1100	3	4	21	37	41	42		
445	43.8505	18.4042	0	1100	3	4	21	37	41	42		
446	43.8511	18.4048	0	240	3	4	21	37	41	42		
447	43.8515	18.4038	0	1100	3	4	21	37	41	42		
448	43.8517	18.4043	0	240	3	4	21	37	41	42		
449	43.8516	18.4054	0	1100	3	4	21	37	41	42		
450	43.8524	18.4045	0	240	3	4	21	37	41	42		
451	43.8522	18.405	0	1100	3	4	21	37	41	42		
452	43.852	18.4057	0	1100	3	4	21	37	41	42		
453	43.8518	18.4063	0	1100	3	4	21	37	41	42		
454	43.85218N	18.4076	0	1100	3	4	21	37	41	42		
455	43.8516	18.4028	0	1100	3	4	21	37	41	42		
456	43.8517	18.4029	0	1100	3	4	21	37	41	42		
457	43.8404	18.381	0	1100	6	7	63	95	111	119	123	126
458	43.842	18.3811	0	1100	3	4	21	37	41	42		
459	43.8464	18.3812	0	1100	3	4	21	37	41	42		
460	43.8466	18.3825	0	1100	3	4	21	37	41	42		
461	43.8457	18.3838	0	1100	3	4	21	37	41	42		
462	43.8451	18.3842	0	1100	3	4	21	37	41	42		
463	43.8459	18.3857	0	1100	3	4	21	37	41	42		

464	43.8457	18.3866	0	1100	3	4	21	37	41	42		
465	43.8423	18.3884	0	1100	3	4	21	37	41	42		
466	43.8421	18.3893	0	1100	3	4	21	37	41	42		
467	43.8456	18.3887	0	1100	3	4	21	37	41	42		
468	43.8471	18.3928	0	1100	3	4	21	37	41	42		
469	43.847	18.3938	0	1100	3	4	21	37	41	42		
470	43.847	18.3944	0	1100	3	4	21	37	41	42		
471	43.8453	18.3914	0	1100	3	4	21	37	41	42		
472	43.8458	18.3937	0	1100	3	4	21	37	41	42		
473	43.845	18.3927	0	1100	3	4	21	37	41	42		
474	43.8463	18.3959	0	1100	6	7	63	95	111	119	123	126
475	43.8454	18.3953	0	1100	3	4	21	37	41	42		
476	43.8429	18.3946	0	1100	3	4	21	37	41	42		
477	43.8425	18.3938	0	1100	3	4	21	37	41	42		
478	43.8426	18.394	0	1100	3	4	21	37	41	42		
479	43.8477	18.397	0	1100	3	4	21	37	41	42		
480	43.8478	18.3978	0	1100	3	4	21	37	41	42		
481	43.8478	18.3987	0	1100	3	4	21	37	41	42		
482	43.8475	18.4	0	1100	3	4	21	37	41	42		
483	43.8493	18.4032	0	1100	3	4	21	37	41	42		
484	43.8501	18.4051	0	1100	3	4	21	37	41	42		
485	43.87	18.3966	0	1100	3	4	21	37	41	42		
486	43.8688	18.3983	0	1100	3	4	21	37	41	42		
487	43.87	18.399	0	1100	3	4	21	37	41	42		
488	43.8685	18.3999	0	1100	3	4	21	37	41	42		
489	43.8685	18.3961	0	1100	3	4	21	37	41	42		
490	43.869	18.3977	0	1100	3	4	21	37	41	42		
491	43.8677	18.3958	0	1100	3	4	21	37	41	42		
492	43.8664	18.4033	0	1100	3	4	21	37	41	42		
493	43.8661	18.4037	0	1100	3	4	21	37	41	42		
494	43.866	18.4058	0	1100	3	4	21	37	41	42		
495	43.8661	18.4061	0	1100	3	4	21	37	41	42		
496	43.8665	18.4064	0	1100	3	4	21	37	41	42		
497	43.8656	18.4077	0	1100	3	4	21	37	41	42		